

LABELSTAR OFFICE

Benutzerhandbuch

Version 4.30 Build 1010
15. Mai 2015

Inhaltsangabe

Labelstar Office



Mit diesem Programm entwerfen und druckern Sie ihre eigenen Etiketten.

- Einfache Bedienung per Drag & Drop
- Unterstützt die gängigsten Barcodetypen
- Direkte Datenbankanbindung möglich
- Individuelles Etikettendesign durch verschiedenste Drucker- und Systemvariablen
- Markups zur flexiblen Textformatierung
- Druckvorschau, Protokollierung, Memory-Card-Unterstützung und weitere Features

Variablen

Variablen dienen dazu, bestimmte Werte, die sich ändern - wie bespielsweise das aktuelle Datum - auf dem Etikett einzufügen.

```
$DateTime ("dd.MM.yyyy HH:mm", UpdateInterval=1, MonthOffset=10)
```

Bestimmte Zeichen innerhalb eines Ausdrucks kennzeichnen und trennen die einzelnen Segmente und ermöglichen eine Zerlegung und Verarbeitung des Ausdrucks.

Die folgende Tabelle beschreibt die reservierten Zeichen.

Zeichen	Bedeutung
$	Kennzeichnet den Beginn einer Variablen. **Hinweis:** Soll das Zeichen direkt verwendet werden muss "$$" eingegeben werden.
(	Kennzeichnet den Beginn der Parameterliste.
)	Kennzeichnet das Ende der Parameterliste.
"	Textkennung
,	Parameter-Trennzeichen
=	Parameter-Wert-Trennzeichen
\	Escapezeichen

Siehe auch

- Systemvariablen
- Druckervariablen

Systemvariablen

Mit Hilfe dieser Variablen können variable Feldinhalte, zur flexiblen Etikettengestaltung, definiert werden.
Systemvariablen werden, im Gegensatz zu Druckervariablen, von der Anwendung verwaltet und berechnet.

Unterstützte Systemvariablen

- Datums- und Uhrzeitvariablen
- Feldvariablen
- Pfadvariablen
- Textvariablen
- Numerator
- Benutzereingabe
- Mathematische Variablen
- Prüfziffernberechnung
- Sonstige Variablen

Datums-/Uhrzeitvariablen

Mit Hilfe dieser Variablen können Datums- und Uhrzeitwerte auf dem Etikett definiert werden.

Unterstützte Datums-/Uhrzeitvariablen

- Aktuelles Datum/Uhrzeit
- Aktuelles Datum
- Aktuelle Uhrzeit
- Datum/Uhrzeit (System)
- Datum/Uhrzeit konvertieren
- Kalenderwoche
- Tag im Jahr
- Wochentag

Aktuelles Datum/Uhrzeit

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt einen Wert zurück, der das aktuelle Datum und die aktuelle Uhrzeit gemäß der Systemeinstellung enthält.

Syntax

```
$CurrentDateTime
```

Rückgabewert

Aktuelles Datum und aktuelle Uhrzeit gemäß der Systemeinstellungen.

Beispiele

```
$CurrentDateTime -> "15.10.2014 11:03:59"
$Format ($CurrentDateTime, "yyMMdd") -> "141015"
$Format ($CurrentDateTime, "hhmmss") -> "110359"
```

Siehe auch

- Aktuelles Datum
- Aktuelle Uhrzeit
- Datum/Uhrzeit (System)
- Datum/Uhrzeit (Drucker)

Aktuelles Datum

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt einen Wert zurück, der das aktuelle Datum gemäß der Systemeinstellung enthält.

Syntax

```
$CurrentDate
```

Rückgabewert

Aktuelles Datum gemäß der Systemeinstellungen.

Beispiele

```
$CurrentDate -> "15.10.2014"
$Format ($CurrentDate, "yyMMdd") -> "141015"
```

Siehe auch

- Aktuelles Datum/Uhrzeit
- Aktuelle Uhrzeit
- Datum/Uhrzeit (System)
- Datum/Uhrzeit (Drucker)

Aktuelle Uhrzeit

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt einen Wert zurück, der die aktuelle Uhrzeit gemäß der Systemeinstellung enthält.

Syntax

`$CurrentTime`

Rückgabewert

Aktuelle Uhrzeit gemäß der Systemeinstellungen.

Beispiele

`$CurrentTime` -> "11:03:59"
`$Format ($CurrentTime, "hhmmss")` -> "110359"

Siehe auch

- Aktuelles Datum/Uhrzeit
- Aktuelle Datum
- Datum/Uhrzeit (System)
- Datum/Uhrzeit (Drucker)

Datum/Uhrzeit (System)

Definiert eine Datums- und Uhrzeitvariable und konvertiert den Wert mit dem angegebenen Format in die entsprechende Zeichenfolgendarstellung.

Syntax

```
$DateTime (format, [Prompt=prompt, UpdateInterval=updateInterval, MonthOffset=monthOffset,
DayOffset=dayOffset, MinOffset=minOffset, StartDate=startDate, Language=language])
```

Parameter

format

Gibt an, wie das Datum und die Uhrzeit formatiert werden soll.

Der *format*-Parameter sollte entweder einen einzelnen Formatbezeichner (siehe Standard-Formatzeichenfolgen für Datum und Uhrzeit) oder ein benutzerdefiniertes Formatmuster (siehe Benutzerdefinierte Formatzeichenfolgen für Datum und Uhrzeit) enthalten, das das Format der zurückgegeben Zeichenfolge definiert. Wenn *format* den Wert **null** hat oder eine leere Zeichenfolge ("") ist, wird der allgemeine Formatbezeichner 'G' verwendet.

prompt (optional, Standard = leer)

Ist ein Eingabeaufforderungstext definiert, wird das Startdatum am Druckbeginn abgefragt.

updateInterval (optional, Standard = 0)

Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
0: Am Druckbeginn
1: Nach jedem Etikett
n: Nach n Etiketten
-1: Nach jedem Datensatzwechsel

monthOffset (optional, Standard = 0)

Monatsoffset (wird zum aktuellen Datum hinzugezählt)

dayOffset (optional, Standard = 0)

Tagesoffset (wird zum aktuellen Datum hinzugezählt)

minOffset (optional, Standard = 0)

Minutenoffset (wird zur aktuellen Uhrzeit hinzugezählt)

startDate (optional, standardmäßig wird das aktuelle Datum und die aktuelle Uhrzeit gemäß der Systemeinstellung verwendet)

Definiert das Startdatum und die Startzeit.

language (optional, standarmäßig wird die unter Windows eingestellte Sprache verwendet)

Sprache, die zur Formatierung der Ausgabe verwendet werden soll. Weitere Informationen finden Sie unter Ländercodes.

Rückgabewert

Formatierter Text.

Beispiele

```
$DateTime ("dd.MM.yyyy") -> "11.09.2013"
$DateTime ("dd.MM.yyyy", StartDate="15.06.2009", MonthOffset=2) -> "15.08.2009"
$DateTime ("D", UpdateInterval=0, DayOffset=2, Language="fr-Fr", StartDate=$ParseDateTime ("131012",
"yyMMdd")) -> "samedi 12 octobre 2013"
$DateTime ("HH:mm:ss") -> "13:20:35"
```

`$DateTime ("hh:mm:ss")` -> "01:20:35"

ID01 = "260514"

`$DateTime ("D", UpdateInterval=0, DayOffset=2, StartDate=$ParseDateTime (<<ID01>>, "ddMMyy"))` -> "Montag, 26. Juni 2014"

Siehe auch

- Aktuelles Datum/Uhrzeit
- Aktuelles Datum
- Aktuelle Uhrzeit
- Datum/Uhrzeit (Drucker)

Datum/Uhrzeit parsen

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Analysiert die angegebene Zeichenfolgendarstellung und konvertiert die Textzeichenfolge in einen Datums- und Uhrzeitwert. Das Format der Zeichenfolgendarstellung muss einem bestimmten Format genau entsprechen. Andernfalls wird ein Fehler ausgegeben.

Syntax

```
$ParseDateTime (text, format, [Language=language])
```

Parameter

text

Eine Zeichenfolge, die die zu konvertierende Datums- und Zeitangabe enthält.

format

Ein Formatbezeichner, der das erforderliche Format von *text* definiert.

Der *format*-Parameter ist eine Zeichenfolge, die entweder einen einzelnen Standardformatbezeichner oder mindestens einen benutzerdefinierten Formatbezeichner enthält, der das erforderliche Format von *text* definiert. Ausführliche Informationen zu gültigen Formatierungscodes finden Sie unter Standard-Formatzeichenfolgen für Datum und Uhrzeit oder Benutzerdefinierte Formatzeichenfolgen für Datum und Uhrzeit.

language (optional, standarmäßig wird die unter Windows eingestellte Sprache verwendet)

Sprache, die angibt welche kulturspezifischen Formatinformationen verwendet werden sollen. Weitere Informationen finden Sie unter Ländercodes.

Rückgabewert

Datums- und Uhrzeitwert.

Beispiele

ID01 = "091410"

```
$ParseDateTime ("130910", "yyMMdd") -> "10.09.2013 00:00:00"
$ParseDateTime (<<ID01>>, "MMyydd") -> "10.09.2014 00:00:00"
```

Siehe auch

- Datum/Uhrzeit (System)

Kalenderwoche

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Berechnet die Kalenderwoche.

Syntax

```
$WeekOfYear (format, [UpdateInterval=updateInterval, MonthOffset=monthOffset, DayOffset=dayOffset,
StartDate=startDate, Language=language])
```

Parameter

format

Gibt an, wie die Kalenderwoche formatiert werden soll.

Formatbezeichner	Beschreibung
w	Kalenderwoche, von 1 bis 53
ww	Kalenderwoche, von 01 bis 53
www	Kalenderwoche, von 001 bis 053
wwww	Kalenderwoche, von 0001 bis 0053
\	Escapezeichen
Jedes andere Zeichen	Das Zeichen wird unverändert in die Ergebniszeichenfolge kopiert

updateInterval (optional, Standard = 0)

Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
0: Am Druckbeginn
1: Nach jedem Etikett
n: Nach n Etiketten
-1: Nach jedem Datensatzwechsel

monthOffset (optional, Standard = 0)

Monatsoffset (wird zum aktuellen Datum hinzugezählt)

dayOffset (optional, Standard = 0)

Tagesoffset (wird zum aktuellen Datum hinzugezählt)

startDate (optional, standardmäßig wird das aktuelle Datum gemäß der Systemeinstellung verwendet)

Definiert das Startdatum.

language (optional, standarmäßig wird die unter Windows eingestellte Sprache verwendet)

Sprache, die zur Formatierung der Ausgabe verwendet werden soll. Weitere Informationen finden Sie unter Ländercodes.

Rückgabewert

Formatierte Kalenderwoche.

Beispiele

Aktuelles Datum: 01.02.2014

```
$WeekOfYear ("w") -> "5"
```

`$WeekOfYear ("ww")` -> "05"
`$WeekOfYear ("www", DayOffset=5)` -> "006"
`$WeekOfYear ("Kalender\woche: ww", StartDate="01.03.2014")` -> "Kalenderwoche: 09"

Tag im Jahr

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Berechnet den Tag im Jahr.

Syntax

```
$DayOfYear (format, [UpdateInterval=updateInterval, MonthOffset=monthOffset, DayOffset=dayOffset,
StartDate=startDate, Language=language])
```

Parameter

format
Gibt an, wie der Tag im Jahr formatiert werden soll.

Formatbezeichner	Beschreibung
d	Tag im Jahr, von 1 bis 366
dd	Tag im Jahr, von 01 bis 366
ddd	Tag im Jahr, von 001 bis 366
dddd	Tag im Jahr, von 0001 bis 0366
\	Escapezeichen
Jedes andere Zeichen	Das Zeichen wird unverändert in die Ergebniszeichenfolge kopiert

updateInterval (optional, Standard = 0)
Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
0: Am Druckbeginn
1: Nach jedem Etikett
n: Nach n Etiketten
-1: Nach jedem Datensatzwechsel

monthOffset (optional, Standard = 0)
Monatsoffset (wird zum aktuellen Datum hinzugezählt)

dayOffset (optional, Standard = 0)
Tagesoffset (wird zum aktuellen Datum hinzugezählt)

startDate (optional, standardmäßig wird das aktuelle Datum gemäß der Systemeinstellung verwendet)
Definiert das Startdatum.

language (optional, standarmäßig wird die unter Windows eingestellte Sprache verwendet)
Sprache, die zur Formatierung der Ausgabe verwendet werden soll. Weitere Informationen finden Sie unter Ländercodes.

Rückgabewert

Formatierter Tag im Jahr.

Beispiele

Aktuelles Datum: 01.02.2014

```
$DayOfYear ("d") -> "5"
```

`$DayOfYear ("dd")` -> "05"
`$DayOfYear ("ddd", DayOffset=5)` -> "006"
`$DayOfYear ("Tag im Jahr: dd", StartDate="01.03.2014")` -> "Tag im Jahr: 09"

Wochentag

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Berechnet den Wochentag

Syntax

```
$DayOfWeek (format, [UpdateInterval=updateInterval, MonthOffset=monthOffset, DayOffset=dayOffset,
StartDate=startDate, Sunday=sunday, Language=language])
```

Parameter

format

Gibt an, wie der Wochentag formatiert werden soll.

Formatbezeichner	Beschreibung
d	Wochentag, von 0 bis 6
dd	Wochentag, von 00 bis 06
ddd	Wochentag, von 000 bis 006
dddd	Wochentag, von 0000 bis 0006
\	Escapezeichen
Jedes andere Zeichen	Das Zeichen wird unverändert in die Ergebniszeichenfolge kopiert

updateInterval (optional, Standard = 0)

Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
0: Am Druckbeginn
1: Nach jedem Etikett
n: Nach n Etiketten
-1: Nach jedem Datensatzwechsel

monthOffset (optional, Standard = 0)

Monatsoffset (wird zum aktuellen Datum hinzugezählt)

dayOffset (optional, Standard = 0)

Tagesoffset (wird zum aktuellen Datum hinzugezählt)

startDate (optional, standardmäßig wird das aktuelle Datum gemäß der Systemeinstellung verwendet)

Definiert das Startdatum.

sunday (optional, Standard = 0)

Definiert, welcher Wert für Sonntag verwendet werden soll.

language (optional, standarmäßig wird die unter Windows eingestellte Sprache verwendet)

Sprache, die zur Formatierung der Ausgabe verwendet werden soll. Weitere Informationen finden Sie unter Ländercodes.

Rückgabewert

Formatierter Wochentag.

Beispiele

Aktuelles Datum: 01.02.2014 (Samstag)

`$DayOfWeek ("d")` -> "6"
`$DayOfWeek ("dd")` -> "06"
`$DayOfWeek ("dd", DayOffset=5)` -> "04"
`$DayOfWeek ("d", DayOffset=5, Sunday=A)` -> "E"
`$DayOfWeek ("Wochentag: d", StartDate="05.03.2014", Sunday=10)` -> "Wochentag: 14"

Feldvariablen

Mit Hilfe der Feldvariablen können Verknüpfungen zwischen einzelnen Elementen auf dem Etikett definiert werden.

Unterstützte Feldvariablen

- Datenbankfeld
- Feldinhalt
- Feldname

Datenbankfeld

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Fügt ein Datenbankfeld auf dem Etikett ein.

Syntax

```
$DbField (dbName, columnName, [DBNullValue=dbNullValue, Format=format])
```

Parameter

dbName
Datenbankname
Hinweis: Groß- und Kleinschreibung wird berücksichtigt

columnName
Spaltenname
Hinweis: Groß- und Kleinschreibung wird berücksichtigt

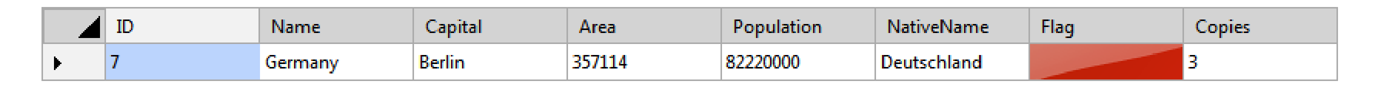
dbNullValue (optional, Standard = leer)
Gibt ab, welcher Wert verwendet werden soll, wenn das zugehörige Datenbankfeld leer ist.

format (optional, Standard = leer)
Gibt an, wie der Inhalt des Datenbankfeldes formatiert werden soll. Weitere Informationen finden Sie unter Formatzeichenfolgen.

Rückgabewert

Inhalt des Datenbankfeldes.

Beispiele



ID	Name	Capital	Area	Population	NativeName	Flag	Copies
7	Germany	Berlin	357114	82220000	Deutschland		3

```
$DbField ("Europe", "Area") -> "357114"
$DbField ("Europe", "Area", Format="0000000000") -> "0000357411"
$DbField ("Europe", "Capital", Format="LLLL") -> "Berl"
$ToUpper ($DbField ("Europe", "Capital")) -> "BERLIN"
```

Überprüfen, ob ein Datenbankfeld leer ist oder nicht

```
$If ($Length ($DBField (...)) == 0, "Das Datenbankfeld ist leer.", "Das Datenbankfeld ist nicht leer.")
```

Siehe auch

- Datenbanken

Feldinhalt auslesen

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den Feldinhalt zurück.

Syntax

```
$FieldLink (fieldName, [displayText])
```

oder

```
<<fieldName>>
```

Parameter

fieldName
Feldname
Hinweis: Groß- und Kleinschreibung wird berücksichtigt

displayText (optional, Standard = leer)
Gibt an, ob ein anderer Wert für die Bildschirmanzeige verwendet werden soll als der eigentliche Feldinhalt.
Hinweis: Für die Druckausgabe wird immer der aktuelle Feldinhalt verwendet.

Rückgabewert

Feldinhalt

Beispiele

ID01 = "12345"
ID02 = "abcABC"

```
$FieldLink (ID01) -> "12345"
$FieldLink (ID01, "00000") -> "00000"
$FieldLink (ID02) -> "abcABC"
$FieldLink (ID02, "XXXXXX") -> "XXXXXX"

<<ID02>> -> "abcABC"
```

Siehe auch

- Kettenfeld (Drucker)

Feldname auslesen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den Feldnamen zurück.

Syntax

```
$FieldName
```

Rückgabewert

Feldname

Pfadvariablen

Mit Hilfe der Pfadvariablen können Pfadzeichenfolgen ausgelesen und bearbeitet werden.

Unterstützte Pfadvariablen

- Anwendungsdatenverzeichnis
- Programmverzeichnis
- Programmpfad
- Verzeichnisname
- Dateierweiterung
- Dateiname
- Grafikverzeichnis
- Etikettenverzeichnis
- Etikettenpfad
- Installationsverzeichnis

$AppDataDir

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt das Verzeichnis zurück, das für programmspezifische Daten verwendet wird, die von allen Benutzern verwendet werden.

Syntax

```
$AppDataDir
```

Rückgabewert

Windows XP: *C:\Dokumente und Einstellungen\All Users\Application Data\Labelstar Office*
Windows 7/8: *C:\ProgramData\Labelstar Office*

$AppDir

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt das aktuelle Programmverzeichnis zurück.

Syntax

```
$AppDir
```

Rückgabewert

Der Pfad für die ausführbare Datei, die die Anwendung gestartet hat (z.B. *C:\Programme\Carl Valentin GmbH\Labelstar Office*).

Siehe auch

- Programmpfad
- Installationsverzeichnis

$AppPath

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den vollständigen Pfadname der Anwendung zurück.

Syntax

```
$AppPath
```

Rückgabewert

Der Pfad und der Name der ausführbaren Datei, die zum Starten der Anwendung verwendet wurde (z.B. *C:\Programme\Carl Valentin GmbH\Labelstar Office\LabelDesigner.exe*).

Siehe auch

- Programmverzeichnis
- Installationsverzeichnis

$Dir

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die Verzeichnisinformationen für die angegebene Pfadzeichenfolge zurück.

Syntax

```
$Dir (path)
```

Parameter

path
: Der Pfad einer Datei oder eines Verzeichnisses.

Rückgabewert

Die Verzeichnisinformationen des angegebenen Pfades oder einen leeren Text, wenn der angegebene Pfad keine Verzeichnisinformationen enthält.

Beispiele

`$Dir ("C:\Labels\Label.lbex") ->` "C:\Labels"
`$Dir ($AppPath) ->` "C:\Programme\Carl Valentin GmbH"

Siehe auch

- Dateiname auslesen
- Dateierweiterung auslesen

$Ext

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die Erweiterung der angegebenen Pfadzeichenfolge zurück.

Syntax

```
$Ext (path)
```

Parameter

path
Die Pfadzeichenfolge, aus der die Erweiterung abgerufen werden soll.

Rückgabewert

Die Erweiterung des angegebenen Pfades (einschließlich ".") oder einen leeren Text (""), wenn der angegebene Pfad keine Erweiterung enthält.

Beispiele

```
$Ext ("C:\label.lbex") -> ".lbex"
$Ext ($AppPath) -> ".exe"
$Ext ("C:\label") -> ""
```

Siehe auch

- Dateiname auslesen
- Verzeichnisname auslesen

$FileName

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den Dateinamen der angegebenen Pfadzeichenfolge zurück.

Syntax

```
$FileName (path, [Ext=extension])
```

Parameter

path

Die Pfadzeichenfolge, aus der der Dateiname abgerufen werden soll.

extension (optional, Standard = true)

true oder 1: Dateiname mit Erweiterung zurückgeben
false oder 0: Dateiname ohne Erweiterung zurückgeben

Rückgabewert

Der Dateiname des angegebenen Pfades mit oder ohne Erweiterung.

Beispiele

```
$FileName ("C:\Labels\Label.lbex") -> "Label.lbex"
$FileName ($AppPath) -> "LabelDesigner.exe"
$FileName ($AppPath, Ext=false) -> "LabelDesigner"
$FileName ($LabelPath) -> "Label.lbex"
$FileName ($LabelPath, Ext=0) -> "Label"
```

Siehe auch

- Verzeichnisname auslesen
- Dateierweiterung auslesen

$ImageDir

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt das aktuelle Grafikverzeichnis zurück.

Syntax

```
$ImageDir
```

Rückgabewert

Aktuelles Grafikverzeichnis

Siehe auch

- Grafikverzeichnis ändern
- Aktuelles Etikettenverzeichnis auslesen

$InstallDir

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt das Verzeichnis zurück, in dem **Labelstar Office** installiert worden ist.

Syntax

```
$InstallDir
```

Rückgabewert

Das Installationsverzeichnis von **Labelstar Office** (z.B. *C:\Programme\Carl Valentin GmbH\Labelstar Office* oder *C:\Programme (x86)\Carl Valentin GmbH\Labelstar Office*).

Siehe auch

- Programmverzeichnis
- Programmpfad

$LabelDir

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt das aktuelle Etikettenverzeichnis zurück.

Syntax

```
$LabelDir
```

Rückgabewert

Aktuelles Etikettenverzeichnis

Siehe auch

- Etikettenverzeichnis ändern
- Aktuelles Grafikverzeichnis auslesen

$LabelPath

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den vollständigen Pfadnamen der aktuellen Etikettendatei zurück.

Syntax

```
$LabelPath
```

Rückgabewert

Pfadname der aktuellen Etikettendatei

Siehe auch

- Aktuelles Grafikverzeichnis auslesen
- Aktuelles Etikettenverzeichnis auslesen

Textvariablen

Mit Hilfe der Textvariablen können Zeichenfolgen bearbeitet werden.

Unterstützte Textvariablen

- Zeichen von Links auslesen
- Zeichen von Rechts auslesen
- Zeichen aus der Mitte auslesen
- Zeichen löschen
- Text ersetzen
- Pattern ersetzen
- Text von Links auffüllen
- Text von Rechts auffüllen
- Text umkehren
- Text in Kleinbuchstaben umwandeln
- Text in Großbuchstaben umwandeln
- Text begrenzen
- Führende Zeichen kürzen
- Nachgestellte Zeichen kürzen
- Führende und nachgestellte Zeichen kürzen
- ASCII-Zeichen in Hexdezimalzeichen umwandeln
- Hexadezimalzeichen in ASCII-Zeichen umwandeln
- Textlänge berechnen
- Text formatieren

Zeichen von Links auslesen

Benötigte Programmvariante	BASIC, PROFESSIONAL

Weitere Informationen finden Sie unter Programmvarianten.

Gibt eine Zeichenfolge zurück, die eine angegebene Zeichenanzahl von der linken Seite einer Zeichenfolge enthält.

Syntax

```
$Left (string, length)
```

Parameter

string

Zeichenfolge, von der die äußersten linken Zeichen zurückgegeben werden sollen.

length

Anzahl Zeichen, die zurückgegeben werden sollen. Ist der Wert 0, wird eine leere Zeichenfolge zurückgegeben. Ist der Wert größer oder gleich der Anzahl von Zeichen in *string*, wird die ganze Zeichenfolge zurückgegeben.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$Left ("abcDEF", 0) -> ""
$Left ("abcDEF", 2) -> "ab"
$Left ("abcDEF", 4) -> "abcD"
$Left ("abcDEF", 10) -> "abcDEF"
$Left ("abcDEF", -2) -> Fehler
```

Siehe auch

- $Mid
- $Right

Zeichen von Rechts auslesen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt eine Zeichenfolge zurück, die eine angegebene Zeichenanzahl von der rechten Seite einer Zeichenfolge enthält.

Syntax

```
$Right (string, length)
```

Parameter

string
: Zeichenfolge, von der die äußersten rechten Zeichen zurückgegeben werden sollen.

length
: Anzahl Zeichen, die zurückgegeben werden sollen. Ist der Wert 0, wird eine leere Zeichenfolge zurückgegeben. Ist der Wert größer oder gleich der Anzahl von Zeichen in *string*, wird die ganze Zeichenfolge zurückgegeben.

Rückgabeparameter

Geänderte Zeichenfolge.

Beispiele

```
$Right ("abcDEF", 0) -> ""
$Right ("abcDEF", 2) -> "EF"
$Right ("abcDEF", 4) -> "cDEF"
$Right ("abcDEF", 10) -> "abcDEF"
$Right ("abcDEF", -2) -> Fehler
```

Siehe auch

- $Left
- $Mid

Zeichen aus der Mitte auslesen

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt eine Zeichenfolge zurück, die eine angegebene Zeichenanzahl aus einer Zeichenfolge enthält.

Syntax

```
$Mid (string, index, [Length=length])
```

Parameter

string
: Zeichenfolge, aus der Zeichen zurückgegeben werden sollen.

index
: Anfangsposition der zurückzugebenden Zeichen. Wenn *index* größer als die Anzahl der Zeichen in *text* ist, wird eine leere Zeichenfolge zurückgegeben.
Hinweis: Die Zeichenposition ist 1-basiert.

length (optional)
: Die Anzahl der zurückzugebenden Zeichen. Wenn der Ausdruck ausgelassen wird oder weniger als *length* Zeichen in *string* enthalten sind (einschließlich des Zeichens an Position *index*), werden alle Zeichen von der Anfangs- bis zur Endposition der Zeichenfolge zurückgegeben.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$Mid ("abcDEF", 3) -> "DEF"
$Mid ("abcDEF", 3, Length=2) -> "DE"
```

Siehe auch

- $Left
- $Right

Zeichen löschen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Löscht die angegebene Anzahl von Zeichen ab der angegebenen Position aus der Zeichenfolge.

Syntax

```
$Remove (string, index, [Length=length])
```

Parameter

string
: Zeichenfolge, die bearbeitet werden soll.

index
: Anfangsposition, ab der die Zeichen gelöscht werden sollen.
Hinweis: Die Zeichenposition ist 0-basiert.

length (optional)
: Die Anzahl der zu löschenden Zeichen. Wird keine Anzahl der zu löschenden Zeichen angegeben, so werden alle Zeichen bis zum Textende gelöscht.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$Remove ("abcDEF", 3) -> "abc"
$Remove ("abcDEF", 3, Length=2) -> "abcF"
```

Text ersetzen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Ersetzt alle Vorkommen einer angegebenen Zeichenfolge durch eine andere angegebene Zeichenfolge.

Syntax

```
$Replace (string, oldValue, newValue, [oldValue, newValue, ...])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

oldValue
Die zu ersetzende Zeichenfolge.

newValue
Die Zeichenfolge, die jedes Vorkommen von *oldValue* ersetzen soll.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$Replace ("abcDEFabcDEF", "abc", "") -> "DEFDEF"
$Replace ("abcDEFabcDEF", "abc", "xxx") -> "xxxDEFxxxDEF"
$Replace ("abcDEFabcDEF", "abc", "xxx", "DEF", "def") -> "xxxdefxxxdef"
```

Siehe auch

- Pattern ersetzen

Pattern ersetzen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Ersetzt in einer angegebenen Eingabezeichenfolge alle mit einem angegebenen regulären Ausdruck übereinstimmenden Zeichenfolgen durch eine angegebene Ersetzungszeichenfolge. Durch angegebene Optionen wird die Suche nach Übereinstimmungen geändert.

Syntax

```
$ReplacePattern (string, pattern/filename, replacement, [IgnoreCase=ignorecase,
RightToLeft=rightToLeft])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

pattern/filename
Das Muster eines regulären Ausdrucks, mit dem Übereinstimmungen gefunden werden sollen oder der Dateiname einer Textdatei, die die Musterdefinition enthält. Die Muster für reguläre Ausdrücke werden in einer speziellen Syntax oder Sprache definiert. Weitere Informationen finden Sie unter Sprachelemente für reguläre Ausdrücke.
Beispiel: Liste der gesuchten Wörter (durch | getrennt) oder der Dateiname einer Textdatei, die die gesuchten Wörter enthält.

replacement
Die Ersatzzeichenfolge.

ignorecase (optional, Standard = false)
true oder 1: Groß-/Kleinschreibung bei der Suche ignorieren
false oder 0: Groß-/Kleinschreibung bei der Suche beachten

righttoleft (optional, Standard = false)
Ändert die Suchrichtung.
true oder 1: Die Suche wird von rechts nach links durchgeführt
false oder 0: Die Suche wird von links nach rechts durchgeführt

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

`$ReplacePattern ("abcdefABCDEF", "abc|DEF", "<b>$0</b>")` -> "**abc**defABC**DEF**"
`$ReplacePattern ("abcdefABCDEF", "abc", "<u><b>$0</b></u>", IgnoreCase=true)` -> "**abc**def**ABC**DEF"

Entfernen von ungültigen Zeichen aus einer Zeichenfolge

In diesem Fall entfernt `$ReplacePattern` alle nicht alphanumerischen Zeichen außer Punkten (.), @-Zeichen und Bindestrichen (-) und gibt die verbleibende Zeichenfolge zurück.

`$ReplacePattern ("<email>@example.com", "[^\w\.@-]", "")` -> "email@example.com"

Weitere Beispiele finden Sie unter Allergenkennzeichnung von Lebensmitteln.

Siehe auch

- Text ersetzen

Sprachelemente für reguläre Ausdrücke

Reguläre Ausdrücke sind Muster, für die eine Entsprechung im Eingabetext gesucht wird. Muster können aus einem oder mehr Zeichenliteralen, Operatoren oder Konstrukten bestehen.

Escapezeichen

Der umgekehrte Schrägstrich (\) in einem regulären Ausdruck gibt an, dass es sich bei dem darauf folgenden Zeichen um ein Sonderzeichen (wie in der folgenden Tabelle angezeigt) handelt oder dass das Zeichen als solches interpretiert werden soll.

Escapezeichen	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
\a	Entspricht einem Klingelzeichen (Warnsignal) \u0007.	`\a`	"Fehler!" + '\u0007'	"\u0007"
\b	Entspricht in einer Zeichenklasse einem Rücktastenzeichen \u0008.	`[\b]{3,}`	"\b\b\b\b"	"\b\b\b\b"
\t	Entspricht einem Tabstoppzeichen \u0009.	`(\w+)\t`	"Element1\tElement2\t"	"Element1\t", "Element2\t"
\r	Entspricht einem Wagenrücklaufzeichen \u000D. (**\r** entspricht nicht dem Zeilenumbruchzeichen **\n**.)	`\r\n(\w+)`	"\r\nDies sind\nzwei Zeilen."	"\r\nDies"
\v	Entspricht einem vertikalen Tabstoppzeichen \u000B.	`[\v]{2,}`	"\v\v\v"	"\v\v\v"
\f	Entspricht einem Seitenvorschubzeichen \u000C.	`[\f]{2,}`	"\f\f\f"	"\f\f\f"
\n	Entspricht einer neuen Zeile \u000A.	`\r\n(\w+)`	"\r\nDies sind\nzwei Zeilen."	"\r\nDies"
\e	Entspricht einem Escapezeichen \u001B.	`\e`	"\x001B"	"\x001B"
**** *nnn*	Verwendet die oktale Darstellung, um ein Zeichen anzugeben (*nnn* besteht aus zwei oder drei Ziffern).	`\w\040\w`	"a bc d"	"a b", "c d"
\x *nn*	Verwendet die hexadezimale Darstellung, um ein Zeichen anzugeben (*nn* besteht genau aus zwei Ziffern).	`\w\x20\w`	"a bc d"	"a b", "c d"
\c *X* **\c** *x*	Entspricht dem durch *X* oder *x* angegebenen ASCII-Steuerzeichen, wobei *X* oder *x* der Buchstabe des Steuerzeichens ist.	`\cC`	"\x0003" (Strg-C)	"\x0003"
\u *nnnn*	Entspricht einem Unicode-Zeichen in hexadezimaler Darstellung (genau vier Stellen, dargestellt durch *nnnn*).	`\w\u0020\w`	"a bc d"	"a b", "c d"
****	Entspricht dem angegebenen Zeichen, wenn darauf ein Zeichen folgt, das in dieser und anderen Tabellen in diesem Thema nicht als Escapezeichen erkannt wird. Beispielsweise ist ***** identisch mit **\x2A** und **\.** entspricht **\x2E**. Hierdurch kann das Modul für reguläre Ausdrücke Sprachelemente (z. B. * oder?) und Zeichenliterale (dargestellt durch * oder \?) unterscheiden.	`\d+[\+-x\*]\d+` `\d+[\+-x\*\d+`	"(2+2) * 3*9"	"2+2", "3*9"

Zeichenklassen

Eine Zeichenklasse entspricht einer beliebigen Reihe von Zeichen. Zeichenklassen verwenden die in der folgenden Tabelle aufgeführten Sprachelemente.

Zeichenklasse	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen

[*character_group* **]**	Entspricht einem beliebigen einzelnen Zeichen in *character_group*. Bei der Entsprechung wird standardmäßig die Groß- und Kleinschreibung berücksichtigt.	`[ae]`	"klasse"	"a", "e"
[^ *character_group* **]**	Negation: Entspricht jedem beliebigen einzelnen Zeichen, das nicht in *character_group* enthalten ist. Standardmäßig wird bei Zeichen in *character_group* die Groß-/Kleinschreibung beachtet.	`[^aei]`	"ringen"	"r", "g", "n"
[*first* **-** *last* **]**	Zeichenbereich: Entspricht jedem beliebigen einzelnen Zeichen im Bereich von *first* bis *last*.	`[A-Z]`	"AB123"	"A", "B"
.	Platzhalterzeichen: Entspricht jedem beliebigen einzelnen Zeichen außer **\n**. Damit es einem Punkt als Literalzeichen entspricht ("." oder **\u002E**), muss ihm ein Escapezeichen (**\.**) vorangestellt werden.	`a.e`	"gerade"	"ade"
\p{ *name* **}**	Entspricht jedem beliebigen Zeichen, das sich in der allgemeinen Unicode-Kategorie oder einem von *name* angegebenen benannten Block befindet.	`\p{IsCyrilli`	"ДЖem"	"Д", "Ж"
\P{ *name* **}**	Entspricht jedem beliebigen Zeichen, das sich nicht in der allgemeinen Unicode-Kategorie oder einem von *name* angegebenen benannten Block befindet.	`\P{IsCyrillic`	"ДЖem"	"e", "m"
\w	Entspricht einem beliebigen Wortzeichen.	`\w`	"ID A1.3"	"I", "D", "A", "1", "3"
\W	Entspricht einem beliebigen Nichtwortzeichen.	`\W`	"ID A1.3"	" ", "."
\s	Entspricht einem beliebigen Leerraumzeichen.	`\w\s`	"ID A1.3"	"D "
\S	Entspricht einem beliebigen Nicht-Leerraumzeichen.	`\s\S`	"int __ctr"	" _"
\d	Entspricht einer beliebigen Dezimalziffer.	`\d`	"4 = IV"	"4"
\D	Entspricht einem beliebigen Zeichen, das keine Dezimalziffer ist.	`\D`	"4 = IV"	" ", "=", " ", "I", "V"

Assertionen

Atomare Assertionen mit einer Breite von Null bewirken, dass, in Abhängigkeit von der Position in der Zeichenfolge, eine Entsprechung gefunden oder nicht gefunden wird.

Assertion	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
^	Der Vergleich muss am Anfang der Zeichenfolge oder Zeile beginnen.	`^\d{3}`	"901-333-"	"901"
$	Der Vergleich muss am Ende der Zeichenfolge oder vor **\n** am Ende der Zeile oder Zeichenfolge erfolgen.	`-\d{3}$`	"-901-333"	"-333"
\A	Der Vergleich muss am Beginn der Zeichenfolge erfolgen.	`\A\d{3}`	"901-333-"	"901"
\Z	Der Vergleich muss am Ende der Zeichenfolge oder vor **\n** am Ende der Zeichenfolge erfolgen.	`-\d{3}\Z`	"-901-333"	"-333"

\z	Der Vergleich muss am Ende der Zeichenfolge erfolgen.	-\d{3}\z	"-901-333"	"-333"
\G	Der Vergleich muss an dem Punkt erfolgen, an dem der vorherige Vergleich beendet wurde.	\G\(\d\)	"(1)(3)(5)[7](9)"	"(1)", "(3)", "(5)"
\b	Der Vergleich muss an einer Begrenzung zwischen einem **\w** (alphanumerischen) und einem **\W** (nicht alphanumerischen) Zeichen erfolgen.	\b\w+\s\w+\b	"dem demnach dem dem"	"dem demnach", "dem dem"
\B	Der Vergleich darf nicht an einer **\b**-Begrenzung erfolgen.	\Bend\w*\b	"end sendet endete sender"	"ends", "ender"

Gruppierungskonstrukte

Gruppierungskonstrukte grenzen Teilausdrücke eines regulären Ausdrucks ab und zeichnen gewöhnlich Teilzeichenfolgen einer Eingabezeichenfolge auf. Gruppierungskonstrukte verwenden die Sprachelemente in der folgenden Tabelle.

Gruppierungskonstrukt	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
(*subexpression* **)**	Zeichnet den übereinstimmenden Teilausdruck auf und weist diesem eine einsbasierte Ordinalzahl zu.	(\w)\1	"paarweise"	"aa"
(?< *name* **>** *subexpression* **)**	Zeichnet den übereinstimmenden Teilausdruck in einer benannten Gruppe auf.	(?<double>\w)\k<double>	"paarweise"	"aa"
(?< *name1* **-** *name2* **>** *subexpression* **)**	Definiert eine Ausgleichsgruppendefinition.	(((?'Open'\()[^\(\)]*)+((?'C Open'\))[^\(\)]*)+)*(?(Open)(?!))$	"3+2^((1-3)*(3-1))"	"((1-3)*(3-1))"
(?: *subexpression* **)**	Definiert eine Nicht-Erfassungsgruppe.	Write(?:Line)	"Console.WriteLine()"	"WriteLine"
(?imnsx-imnsx: *subexpression* **)**	Aktiviert oder deaktiviert die angegebenen Optionen in *subexpression*.	A\d{2}(?i:\w+)\b	"A12xl A12XL a12xl"	"A12xl", "A12XL"
(?= *subexpression* **)**	Positive Lookaheadassertion mit einer Breite von Null.	\w+(?=\.)	"Er isst. Der Hund rennt. Die Sonne ist weg."	"isst", "rennt", "weg"
(?! *subexpression* **)**	Negative Lookaheadassertion mit einer Breite von Null.	\b(?!un)\w+\b	"unsicher sicher unter mischt"	"sicher", "mischt"
(?<= *subexpression* **)**	Positive Lookbehindassertion mit einer Breite von Null.	(?<=19)\d{2}\b	"1851 1999 1950 1905 2003"	"99", "50", "05"
(?<! *subexpression* **)**	Negative Lookbehindassertion mit einer Breite von Null.	(?<!19)\d{2}\b	"1851 1999 1950 1905 2003"	"51", "03"
(?> *subexpression* **)**	Nicht zurückverfolgender ("gieriger") Teilausdruck.	[13579](?>A+B+)	"1ABB 3ABBC 5AB 5AC"	"1ABB", "3ABB", "5AB"

Quantifizierer

Quantifizierer geben an, wie viele Instanzen des vorherigen Elements (bei dem es sich um ein Zeichen, eine Gruppe oder eine Zeichenklasse handeln kann) in der Eingabezeichenfolge vorhanden sein müssen, damit eine Entsprechung gefunden wird. Quantifizierer verwenden die Sprachelemente in der folgenden Tabelle.

Quantifizierer	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
*****	Entspricht dem vorangehenden Element nicht oder mehrmals.	\d*\. \d		".0", "19.9", "219.9"

+	Entspricht dem vorangehenden Element einmal oder mehrmals.	"be+"	"beere" "bei"	"bee" "be"
?	Entspricht dem vorangehenden Element nicht oder einmal.	"rai? n"		"ran", "rain"
{ n **}**	Entspricht dem vorangehenden Element genau n-mal.	",\d{3}"	"1,043.6" "9,876,543,210"	",043" ",876", ",543", ",210"
{ n **,}**	Entspricht dem vorangehenden Element mindestens n-mal.	"\d{2,}"		"166", "29", "1930"
{ n **,** m **}**	Entspricht dem vorangehenden Element mindestens n-, höchstens jedoch m-mal.	"\d{3,5}"	"193024"	"19302"
***?**	Entspricht dem vorangehenden Element nicht oder mehrmals, jedoch so wenige Male wie möglich.	\d*? \. \d		".0", "19.9", "219.9"
+?	Entspricht dem vorangehenden Element ein- oder mehrmals, jedoch so wenige Male wie möglich.	"be+?"	"bei"	"be"
??	Entspricht dem vorangehenden Element nicht oder einmal, jedoch so wenige Male wie möglich.	"rai?? n"		"ran", "rain"
{ n **}?**	Entspricht dem vorangehenden Element genau n-mal.	",\d{3}?"	"1,043.6" "9,876,543,210"	",043" ",876", ",543", ",210"
{ n **,}?**	Entspricht dem vorangehenden Element mindestens n-mal, jedoch so wenige Male wie möglich.	"\d{2,}?"		"166", "29", "1930"
{ n **,** m **}?**	Entspricht dem vorangehenden Element zwischen n- und m-mal, jedoch so wenige Male wie möglich.	"\d{3,5}?"	"193024"	"193", "024"

Rückverweiskonstrukte

Ein Rückverweis ermöglicht es, einen zuvor gefundenen Teilausdruck später im gleichen regulären Ausdruck zu identifizieren. In der folgenden Tabelle sind die Rückverweiskonstrukte aufgeführt, die von regulären Ausdrücken unterstützt werden.

Rückverweiskonstrukt	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
**** *number*	Rückverweis. Entspricht dem Wert eines nummerierten Teilausdrucks.	(\w)\1	"beseelt"	"ee"
\k< *name* **>**	Benannter Rückverweis. Entspricht dem Wert eines benannten Ausdrucks.	(?<char> \w)\k<char>	"beseelt"	"ee"

Alternierungskonstrukte

Alternierungskonstrukte ändern einen regulären Ausdruck, um entweder/oder-Vergleiche zuzulassen. Diese Konstrukte verwenden die Sprachelemente in der folgenden Tabelle.

Alternierungskonstrukt	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
|	Entspricht jedem beliebigen durch einen senkrechten Strich (|) getrennten Element.	th(e|is|at)	"This is the day. "	"the", "this"
(?(*expression* **)** *yes* **|** *nein* **)**	Entspricht *yes*, wenn das von *expression* angegebene Muster für reguläre Ausdrücke übereinstimmt. Andernfalls entspricht es dem optionalen *no*.	(?(A)A\d{2}\b|\b \d{3}\b)	"A10 C103 910"	"A10", "910"

	expression wird als Assertion mit einer Breite von Null interpretiert.			
(?(*name* **)** *yes* **|** *no* **)**	Entspricht *yes*, wenn *name*, eine benannte oder nummerierte Erfassungsgruppe, eine Übereinstimmung aufweist. Andernfalls entspricht es dem optionalen *no*.	`(?<quoted>")?(?(quoted).+?"|\S+\s)`	"Hund.jpg "Yiska spielt.jpg""	"Hund.jpg", ""Yiska spielt.jpg""

Ersetzungen

Ersetzungen sind Sprachelemente regulärer Ausdrücke, die in Ersetzungsmustern unterstützt werden. Die Metazeichen in der folgenden Tabelle sind atomare Assertionen mit einer Breite von Null.

Zeichen	Beschreibung	Muster	Ersetzungsmuster	Eingabezeichenfolge	Ergebniszeichenfolge
$ *number*	Ersetzt die untergeordnete Zeichenfolge, die der *number* einer Gruppe entspricht.	`\b(\w+)(\s)(\w+)\b`	`$3$2$1`	"one two"	"two one"
${ *name* **}**	Ersetzt die untergeordnete Zeichenfolge, die dem genannten *name* der Gruppe entspricht.	`\b(?<word1>\w+)(\s)(?<word2>\w+)\b`	`${word2} ${word1}`	"one two"	"two one"
$$	Ersetzt ein "$"-Literal.	`\b(\d+)\s?USD`	`$$$1`	"103 USD"	"$103"
$&	Ersetzt eine Kopie der gesamten Entsprechung.	`\$? \d*\.?\d+`	`**$&**`	"$1.30"	"**$1.30**"
$`	Ersetzt den gesamten Text der Eingabezeichenfolge vor der Entsprechung.	`B+`	`` $` ``	"AABBCC"	"AAAACC"
$'	Ersetzt den gesamten Text der Eingabezeichenfolge nach der Entsprechung.	`B+`	`$'`	"AABBCC"	"AACCCC"
$+	Ersetzt die zuletzt erfasste Gruppe.	`B+(C+)`	`$+`	"AABBCCDD"	AACCDD
$_	Ersetzt die gesamte Eingabezeichenfolge.	`B+`	`$_`	"AABBCC"	"AAAABBCCCC"

Optionen für reguläre Ausdrücke

Sie können Optionen angeben, die steuern, wie ein Muster des regulären Ausdrucks interpretiert wird.

Sie können eine Option auf zwei Arten angeben:

- Mit **(?imnsx-imnsx)**, einem der verschiedenen Konstrukte, bei dem ein Minuszeichen (-) vor einer Option oder einem Optionensatz diese Optionen deaktiviert. Zum Beispiel aktiviert **(?i-mn)** Übereinstimmungen ohne Berücksichtigung der Groß-/Kleinschreibung **(i)**, deaktiviert Mehrzeilenmodus **(m)** und deaktiviert unbenannte Gruppenerfassungen **(n)**. Die Option gilt für das Muster des regulären Ausdrucks ab dem Punkt, an dem die Option definiert ist, und ist entweder bis zum Ende des Musters oder bis zu dem Punkt gültig, an dem ein anderes Konstrukt die Option umkehrt.
- Mit dem Gruppierungskonstrukt **(?imnsx-imnsx:***subexpression***)**, das die Optionen nur für die angegebene Gruppe definiert.

Folgende Optionen werden unterstützt:

Option	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen

i	Groß-/Kleinschreibung bei der Suche ignorieren.	`\b(?i)a(?-i)a\w+\b`	"Aale essen Aasblumen roh"	"Aale", "Aasblumen"
m	Mehrzeilenmodus verwenden. **^** und **$** entsprechen dem Anfang und Ende einer Zeile anstatt dem Anfang und Ende einer Zeichenfolge.			
n	Unbenannte Gruppen nicht erfassen.			
s	Einzeilenmodus verwenden.			
x	Leerraum ohne Escapezeichen im Muster eines regulären Ausdrucks ignorieren.	`\b(?x) \d+ \s \w+`	"1 Erdferkel 2 Katzen IV Zenturionen"	"1 Erdferkel", "2 Katzen"

Verschiedene Konstrukte

Verschiedene Konstrukte ändern Muster von regulären Ausdrücken oder stellen Informationen darüber bereit. In der folgenden Tabelle sind die verschiedenen Konstrukte aufgeführt.

Konstrukt	Beschreibung	Muster	Eingabezeichenfolge	Entsprechungen
(?imnsx-imnsx)	Aktiviert oder deaktiviert Optionen wie die Groß-/Kleinschreibung mitten in einem Muster. Weitere Informationen finden Sie unter Optionen für reguläre Ausdrücke.	\bA(?i)b\w+\b	"ABA Able Act"	"ABA", "Able"
(?# *comment* **)**	Inlinekommentar. Der Kommentar endet bei der ersten schließenden Klammer.	\bA(?#Matches words starting with A)\w+\b		
# [bis Zeilenende**]**	X-Modus-Kommentar. Der Kommentar beginnt bei einem # ohne Escapezeichen und reicht bis zum Ende der Zeile.	(?x)\bA\w+\b#Matches words starting with A		

Text von Links auffüllen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Richtet die Zeichenfolge rechtsbündig aus, indem die linke Seite mit einem angegebenen Zeichen aufgefüllt wird, um die angegebene Gesamtlänge zu erreichen. Wird kein Zeichen angegeben so wird mit Leerzeichen aufgefüllt.

Syntax

```
$PadLeft (string, totalWidth, [PaddingChar=paddingChar])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

totalWidth
Die Anzahl der Zeichen in der resultierenden Zeichenfolge, entsprechend der Anzahl der ursprünglichen Zeichen zuzüglich aller zusätzlichen Füllzeichen.

paddingChar (optional, Standard = Leerzeichen)
Füllzeichen.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$PadLeft ("abcDEF", 10) -> "    abcDEF"
$PadLeft ("12345", 10, PaddingChar="0") -> "0000012345"
```

Siehe auch

- Text von Rechts auffüllen

Text von Rechts auffüllen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Richtet die Zeichenfolge linksbündig aus, indem die rechte Seite mit einem angegebenen Zeichen aufgefüllt wird, um die angegebene Gesamtlänge zu erreichen. Wird kein Füllzeichen angegeben so wird mit Leerzeichen aufgefüllt.

Syntax

```
$PadRight (string, totalWidth, [PaddingChar=paddingChar])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

totalWidth
Die Anzahl der Zeichen in der resultierenden Zeichenfolge, entsprechend der Anzahl der ursprünglichen Zeichen zuzüglich aller zusätzlichen Füllzeichen.

paddingChar (optional, Standard = Leerzeichen)
Füllzeichen.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$PadRight ("abcDEF", 10) -> "abcDEF    "
$PadRight ("12345", 10, PaddingChar="0") -> "1234500000"
```

Siehe auch

- Text von Links auffüllen

Text umkehren

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Kehrt die Reihenfolge der Zeichen in der angegebenen Zeichenfolge um.

Syntax

```
$Reverse (string)
```

Parameter

string

Zeichenfolge, die bearbeitet werden soll.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$Reverse ("abcDEF") -> "FEDcba"
$Reverse ("12345") -> "54321"
```

Text in Kleinbuchstaben umwandeln

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Konvertiert alle Buchstaben der Zeichenfolge in Kleinbuchstaben.

Syntax

```
$ToLower (string)
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

`$ToLower ("abcDEF")` -> "abcdef"

Siehe auch

- Zeichenfolge in Großbuchstaben umwandeln

Text in Großbuchstaben umwandeln

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Konvertiert alle Buchstaben der Zeichenfolge in Großbuchstaben.

Syntax

```
$ToUpper (string)
```

Parameter

string

Zeichenfolge, die bearbeitet werden soll.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$ToUpper ("abcDEF") -> "ABCDEF"
```

Siehe auch

- Zeichenfolge in Kleinbuchstaben umwandeln

Text begrenzen

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Schneidet eine Zeichenfolge ab, wenn sie zu lang ist.

Syntax

```
$Truncate (string, maxLength)
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

maxLength
Maximale Anzahl Zeichen (einschließlich Auslassungszeichen "..."). Ist die Anzahl von Zeichen in *string* größer als *maxLength*, so wird die Zeichenfolge abgeschnitten und "..." angehängt.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$Truncate ("Beispieltext", 8) -> "Beisp..."
$Truncate ($LabelPath, 20) -> "C:\...\Label1.lbex"
```

Führende Zeichen kürzen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Entfernt alle führenden Leerzeichen oder Vorkommen der Zeichen in *trimChars* aus der aktuellen Zeichenfolge.

Syntax

```
$TrimLeft (string, [TrimChars=trimChars])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

trimChars (optional, Standard = Leerzeichen)
Zeichen, die entfernt werden sollen.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$TrimLeft ("   abcDEF   ") -> "abcDEF "
$TrimLeft ("abcDEF", TrimChars="a") -> "bcDEF"
```

Siehe auch

- Führende und nachgestellte Zeichen löschen
- Nachgestellte Zeichen löschen

Nachgestellte Zeichen kürzen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Entfernt alle nachgestellten Leerzeichen oder Vorkommen der Zeichen in *trimChars* aus der aktuellen Zeichenfolge.

Syntax

```
$TrimRight (string, [TrimChars=trimChars])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

trimChars (optional, Standard = Leerzeichen)
Zeichen, die entfernt werden sollen.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

```
$TrimRight ("  abcDEF  ") -> " abcDEF"
$TrimRight ("abcDEF", TrimChars="F") -> "abcDE"
```

Siehe auch

- Führende und nachgestellte Zeichen löschen
- Führende Zeichen löschen

Führende und nachgestellte Zeichen kürzen

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Entfernt alle führenden und nachgestellten Leerzeichen oder Vorkommen der Zeichen in *trimChars* aus der aktuellen Zeichenfolge.

Syntax

```
$Trim (string, [TrimChars=trimChars])
```

Parameter

string
Zeichenfolge, die bearbeitet werden soll.

trimChars (optional, Standard = Leerzeichen)
Zeichen, die entfernt werden sollen.

Rückgabewert

Geänderte Zeichenfolge.

Beispiele

`$Trim ("  abcDEF  ")` -> "abcDEF"
`$Trim ("abcDEF", TrimChars="aF")` -> "bcDE"

Siehe auch

- Führende Zeichen löschen
- Nachgestellte Zeichen löschen

ASCII-Zeichen in Hexdezimalzeichen umwandeln

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Konvertiert eine ASCII-Zeichenfolge in eine Hex-Zeichenfolge.

Syntax

```
$StringToHex (string)
```

Parameter

string

ASCII-Zeichenfolge, die konvertiert werden soll.
Hinweis: Jedes einzelne Zeichen wird in einen zweistelligen Hexadezimalwert umgewandelt.

Rückgabewert

Hex-Zeichenfolge.

Beispiele

```
$StringToHex ("12345") -> "3132333435"
$StringToHex ("abcXYZ") -> "61626358595A"
```

Siehe auch

- Hexadezimalzeichen in ASCII-Zeichen umwandeln

Hexadezimalzeichen in ASCII-Zeichen umwandeln

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Konvertiert eine Hex-Zeichenfolge in eine ASCII-Zeichenfolge.

Syntax

```
$HexToString (string)
```

Parameter

string

Hex-Zeichenfolge, die konvertiert werden soll.
Hinweis: Ein einzelner Hexadezimalwert besteht immer aus zwei Stellen und kann nur Ziffern (0-9) und Buchstabe (a-f, A-F) enthalten.

Rückgabewert

ASCII-Zeichenfolge.

Beispiele

```
$HexToString ("3132333435") -> "12345"
$HexToString ("61626358595A") -> "abcXYZ"
```

Siehe auch

- ASCII-Zeichen in Herxadezimalzeichen umwandeln

Textlänge berechnen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die Anzahl der Zeichen in der Zeichenfolge zurück.

Syntax

```
$Length (string)
```

Parameter

string
Die Zeichenfolge, deren Länge berechnet werden soll.

Rückgabewert

Länge der Zeichenfolge

Beispiele

```
$Length ("abcDEF") -> 6
$Length ("") -> 0
```

Überprüfen, ob ein Datenbankfeld leer ist oder nicht

```
$If ($Length ($DBField (...)) == 0, "Das Datenbankfeld ist leer.", "Das Datenbankfeld ist nicht leer.")
```

Numerator (System)

Fügt einen Numerator auf dem Etikett ein.

Syntax

```
$Counter (value, [Prompt=prompt, UpdateInterval=updateInterval, Increment=increment,
MinValue=minValue, MaxValue=maxValue, TrimLeft=trimLeft, Mode=mode, Radix=radix])
```

Parameter

value

Aktueller Startwert.
Hinweis: Die Anzahl der Stellen legt das Ausgabeformat fest (maximal "999999999").

prompt (optional, Standard = leer)

Ist ein Eingabeaufforderungstext definiert, wird der Startwert am Druckbeginn abgefragt.

updateInterval (optional, Standard = 1)

Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
1: Nach jedem Etikett
n: Nach n Etiketten
-1: Nach jedem Datensatzwechsel

increment (optional, Standard = 1)

Schrittweite.

minValue (optional, Standard = leer)

Minimaler Wert. Wird kein *minValue* angegeben wird standardmäßig die Stellenanzahl des Startwertes verwendet um eine Minimalwert zu berechnen.

Startwert	Basis	Berechneter Minimalwert
0001	10	0000
001A	16	0000
ABC	1	AAA

maxValue (optional, Standard = leer)

Maximaler Wert. Wird kein *maxValue* angegeben wird standardmäßig die Stellenanzahl des Startwertes verwendet um einen Maximalwert zu berechnen.

Startwert	Basis	Berechneter Maximalwert
0001	10	9999
001A	16	FFFF
ABC	1	ZZZ

trimLeft (optional, Standard = false)

true oder 1: Führende Nullen bei der Ausgabe unterdücken
false oder 0: Führnde Nullen bei der Ausgabe anzeigen

mode (optional, Standard = 3)

Betriebsart
0: Startwert am Druckbeginn zurücksetzen
1: Startwert am Druckbeginn zurücksetzen (automatischer Überlauf)
2: Startwert manuell zurücksetzen

3: Startwert manuell zurücksetzen (automatischer Überlauf)

radix (optional, Standard = 10)
Radix, Basis des Numerators (1-36)
1: Alphabetisch (A-Z)
2: Binär (0, 1)
8: Oktal (0-7)
10: Dezimal (0-9)
16: Hexadezimal (0-9, A-F)
36: Alphanumerisch (0-9, A-Z)

Rückgabewert

Aktueller Numeratorwert.

Beispiele

`$Counter ("0001", MinValue="0000", MaxValue="0009", Increment=1, Radix=10)` -> 0001, 0002, 0003, 0004, 0005, 0006, 0007, 0008, 0009, 0009, 0009, ...
`$Counter ("0001", Increment=1, TrimLeft=true, Radix=10)` -> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...
`$Counter ("0001", MinValue="0000", MaxValue="0009", Increment=-1, StartMode=0, Radix=10)` -> 0009, 0008, 0007, 0006, 0005, 0004, 0003, 0002, 0001, 0000, 0000, 0000, ...
`$Counter ("0001", MinValue="0000", MaxValue="0009", Increment=1, StartMode=1, Radix=10)` -> 0000, 0001, 0002, 0003, 0004, 0005, 0006, 0007, 0008, 0009, 0000, 0001, ...

Druckanzahl = "200"

`$Counter ($Copies, Increment=-1, Radix=10)` -> 200, 199, 198, 197, 196, 195, 194, 193, 192, 191, 190, ...

Hexadezimaler Numerator

`$Counter ("0009", MinValue="0000", MaxValue="FFFF", Increment=1, Radix=16)` -> 0009, 000A, 000B, 000C, 000D, 000E, 000F, 0010, 0011, 0012, ...

Siehe auch

- Globaler Numerator
- Numerator (Drucker)

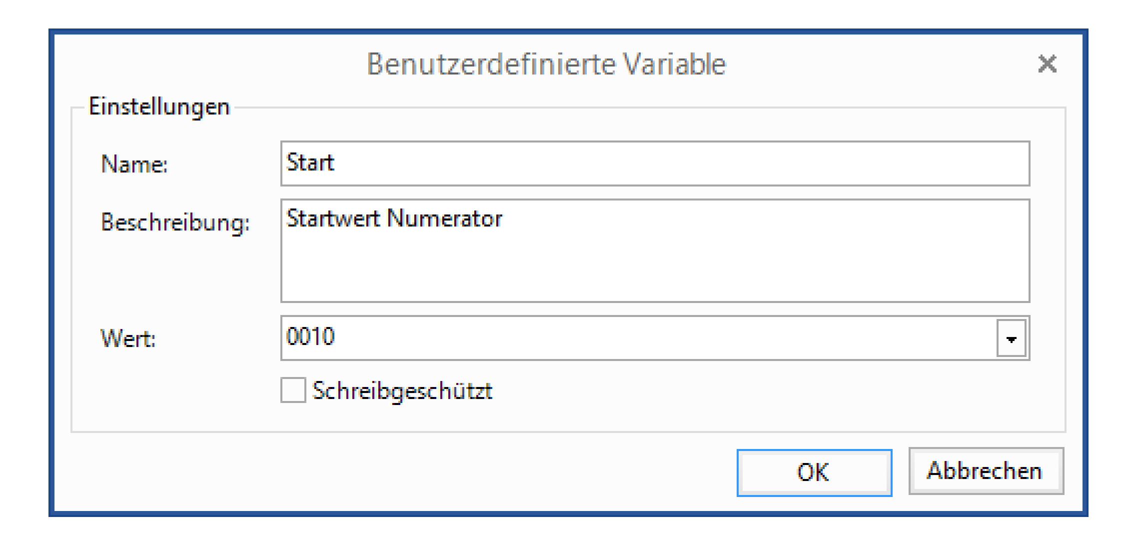
Globaler Numerator

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Der globale Numerator ist der Sonderfall eines Numerators (System). Dabei wird der Startwert global, d.h. etikettenübergeifend definiert und gespeichert.

Um einen globalen Numerator zu definieren, gehen Sie bitte folgendermaßen vor:

1. Wählen Sie einen Text oder Barcode aus.
2. Öffnen Sie den **Variablen-Editor**.
3. Legen Sie eine neue benutzerdefinierte Variable an.



4. Definieren Sie einen Numerator (System) und fügen Sie als Startwert die neu definierte Variable ein.

Beispiele

Start = "0010"

`$Counter ($Start, MinValue="0001", MaxValue="0010", Increment=1, Radix=10)` -> 0010, 0001, 0002, 0003, 0004, 0005, ...

Benutzereingabe (System)

Fügt eine Benutzereingabe (System) auf dem Etikett ein.

Syntax

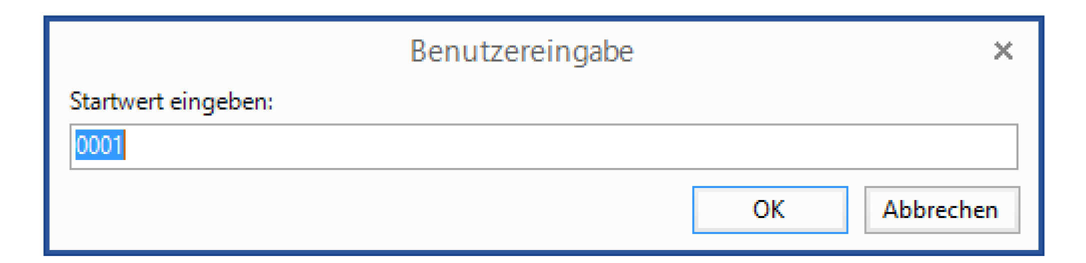
```
$UserInput
```

Rückgabewert

Der eingegebene Text.

Beispiele

Eingabeaufforderungstext = "Startwert eingeben:"
Starttext = "0001"



Siehe auch

- [Benutzereingabe (Drucker)]

Eingabemaske

Die Eingabemaske muss eine Zeichenfolge sein, die entsprechend den Angaben in der folgenden Tabelle aus einem oder mehreren Maskierungselementen besteht.

Maskierungselement	Beschreibung
0	Ziffer (Eingabe erfoderlich)
9	Ziffer oder Leerzeichen (Eingabe optional)
#	Ziffer/Leerzeichen/+/- (Eingabe optional)
L	Buchstabe (Eingabe erforderlich)
?	Buchstabe (Eingabe optional)
A	Buchstabe oder Ziffer (Eingabe erforderlich)
a	Buchstabe oder Ziffer (Eingabe optional)
&	Beliebige Zeichen (Eingabe erforderlich)
C	Beliebige Zeichen (Eingabe optional)
.	Dezimaltrennzeichen
,	Tausendertrennzeichen
:	Trennzeichen für Zeitangaben
/	Trennzeichen für Datumsangaben
$	Währungssymbol
<	In Kleinbuchstaben umwandlen. Konvertiert alle nachfolgenden Zeichen in Kleinbuchstaben.
>	In Großbuchstaben umwandeln. Konvertiert alle nachfolgenden Zeichen in Großbuchstaben.
|	Deativiert die vorangegangene Umwandlung in Klein- oder Großbuchstaben.
!	Ausgabe von rechts nach links. Gibt alle nachfolgenden Zeichen in umgekehrter Reichenfolge aus.
^	Deaktiviert die vorangegangene Umkehrung der Ausgabereihenfolge.
\	Escape. Wandelt ein Maskenzeichen in ein Literal um.
Alle anderen Zeichen	Literale. Alle nicht maskierten Elemente werden in ihrer ursprünglichen Form angezeigt. Literale nehmen zur Laufzeit immer eine statische Position in der Eingabemaske ein und können weder verschoben noch gelöscht werden.

Beispiele:

Format	Verhalten
00/00/0000	Ein Datum (Tag, numerischer Monat, Jahr) im internationalen Datumsformat. Das "/"-Zeichen ist ein logisches Datumstrennzeichen. Dem Benutzer wird dafür das entsprechende Datumstrennzeichen der aktuellen Kultur der Anwendung angezeigt.
00->L<LL-0000	Ein Datum (Tag, Monatsabkürzung und Jahr) im US-Format, in dem die aus drei Buchstaben bestehende Abkürzung für den Monat mit einem großen Anfangsbuchstaben gefolgt von zwei Kleinbuchstaben angezeigt wird.
(999)-000-0000	US-Telefonnummer, optionale Ortskennzahl. Wenn der Benutzer die optionalen Zeichen nicht eingeben möchten, können entweder Leerzeichen eingegeben werden, oder der Mauszeiger kann direkt an der Position in der Maske platziert werden, die durch die erste 0 (null) dargestellt wird.
$999,999.00	Ein Währungswert im Bereich von 0 bis 999999. Die Zeichen für Währungs-, Tausender- und Dezimalzeichen werden durch kulturspezifische Entsprechungen ersetzt.

Mathematische Variablen

Mit Hilfe dieser Variablen können Zahlen bearbeitet und mathematische Formeln berechnet werden.

Unterstützte mathematische Variablen

- Absolutwert
- Minimalwert
- Maximalwert
- Mathematisches Formel berechnen

Absolutwert

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den Absolutbetrag einer Zahl zurück.

Syntax

```
$Abs (value)
```

Parameter

value
: Eine Zahl.

Rückgabewert

Absolutbetrag der angegebenen Zahl.

Beispiele

`$Abs (12.00)` -> "12"
`$Abs (-12.25)` -> "12.25"
`$Format ($Abs (-144), "00000")` -> "00144"

Minimalwert

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die kleinere von zwei Zahlen zurück.

Syntax

```
$Min (value1, value2)
```

Parameter

value1
: Eine Zahl.

value2
: Eine Zahl.

Rückgabewert

Die kleinere der angegebenen Zahlen.

Beispiele

```
$Min (10, 20) -> "10"
$Min (10, 5) -> "5"
```

Siehe auch

- $Max

Maximalwert

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die größere von zwei Zahlen zurück.

Syntax

```
$Max (value1, value2)
```

Parameter

value1
: Eine Zahl.

value2
: Eine Zahl.

Rückgabewert

Die größerer der angegebenen Zahlen.

Beispiele

```
$Max (10, 20) -> "20"
$Max (10, 5) -> "10"
```

Siehe auch

- $Min

Mathematische Formel berechnen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert ein mathematisches Kettenfeld.

Syntax

```
$MathField (formula)
```

Parameter

formula
Formel, die berechnet werden soll. Weitere Informationen finden Sie unter Mathematische Operatoren.

Rückgabewert

Berechneter Wert.

Beispiele

ID01 = "-10.00"
ID02 = "12.00"

```
$MathField (12 * 12) -> "144"
$MathField (<<ID01>> + <<ID02>>) -> "2"
$MathField ($Abs (<<ID01>>) + <<ID02>>) -> "22"

$MathField ((12 * 12) / 10) -> "14,4"
$Format ($MathField ((12 * 12) / 10), "0.00") -> "14,40"
$Format ($MathField ((12 * 12) / 10), "0") -> "14"
```

	ID	Name	Capital	Area	Population	NativeName	Flag	Copies
▸	7	Germany	Berlin	357114	82220000	Deutschland		3

```
$MathField ($DbField ("Europe", "Population") * 2.00) -> "164440000"
```


Mathematische Operatoren

Ein Operator ist ein Term oder ein Symbol, dem ein oder mehrere Ausdrücke bzw. Operanden als Eingabe übergeben werden, und der einen Wert zurückgibt.

Unäre Operatoren (Operatoren mit einem Operanden)

Ausdruck	Beschreibung
+x	Identität
-x	Negation
!x	Logische Negation

Arithmetische Operatoren

Ausdruck	Beschreibung
x + y	Addition, Zeichenfolgenverkettung
x - y	Subtraktion
x * Y	Multiplikation
x / y	Division
x % y	Rest (Modulo)
x ^ y	Potenziert eine angegebene Zahl mit dem angegebenen Exponenten.

Vergleichsoperatoren

Ausdruck	Beschreibung
x = y oder x == y	Gleich
x != y oder x <> y	Ungleich
x < y	Kleiner als
x <= y	Kleiner oder gleich
x > y	Größer als
x >= y	Größer oder gleich

Logische Operatoren

Ausdruck	Beschreibung
x && y	Bedingtes Und. Wertet y nur aus, wenn x den Wert true hat.
x || y	Bedingtes Oder. Werte y nur aus, wenn x den Wert false hat.

Prüfziffernberechnung

Mit Hilfe dieser Variablen können Prüfziffern berechnet werden.

Unterstützte Variablen

- Prüfziffer (System)
- Prüfziffer anhängen

Prüfziffer (System)

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Berechnet die Prüfziffer für die vorgegebene Nutzzeichenfolge.

Syntax

```
$CheckDigit (data, checkDigitMethod)
```

Parameter

data
Nutzzeichenfolge, für die die Prüfsumme berechnet werden soll.

checkDigitMethod
Methode, nach der die Prüfziffer berechnet werden soll.

Methode	Beschreibung
MOD10	Modulo 10
MOD10_LUHN	Modulo 10 (Luhn-Algorithmus)
MOD11	Modulo 11
MOD43	Modulo 43
MOD47_15	Modulo 47 (Gewichtung 15)
MOD47_20	Modulo 47 (Gewichtung 20)
MOD103	Modulo 103

Rückgabewert

Berechnete Prüfziffer.

Beispiele

NVE = "340123451234567 89"

```
$CheckDigit ("12345", MOD10) -> 7
$CheckDigit (<<NVE>>, MOD10) -> 5
```

Siehe auch

- Prüfziffer anhängen
- Prüfziffer (Drucker)

Prüfziffer anhängen

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Berechnet die Prüfziffer für die vorgegebene Nutzzeichenfolge.

Syntax

```
$AppendCheckDigit (data, checkDigitMethod, [AppendTo=appendTo])
```

Parameter

data
Nutzzeichenfolge, für die die Prüfsumme berechnet werden soll.

checkDigitMethod
Methode, nach der die Prüfziffer berechnet werden soll.

Methode	Beschreibung
MOD10	Modulo 10
MOD10_LUHN	Modulo 10 (Luhn-Algorithmus)
MOD11	Modulo 11
MOD43	Modulo 43
MOD47_15	Modulo 47 (Gewichtung 15)
MOD47_20	Modulo 47 (Gewichtung 20)
MOD103	Modulo 103

appendTo (optional, Standard = Right)
Gibt an, wo die berechnete Prüfziffer an die Nutzzeichenfolge angehängt werden soll.
Left: Gibt an, dass die Prüfziffer am Anfang der Nutzzeichenfolge eingefügt wird.
Right: Gibt an, dass die Prüfziffer am Ende der Nutzzeichenfolge angehängt wird.

Rückgabewert

Nutzzeichenfolge mit angehängter Prüfziffer.

Beispiele

NVE = "34012345123456789"

```
$AppendCheckDigit ("12345", MOD10) -> 123457
$AppendCheckDigit (<<NVE>>, MOD10) -> 3401234512345678955
$AppendCheckDigit (<<NVE>>, MOD10, AppendTo=Left) -> 5340123451234567890
```

Siehe auch

- Prüfziffer (System)

Sonstige Variablen

Mit Hilfe dieser Variablen können verschiedene Informationen auf dem Etikett definiert werden.

Unterstützte Variablen

- Druckanzahl
- If..Then..Else-Anweisung
- Schichtdefinition
- Etikettennummer
- Seitennummer
- Druckername
- Benutzername
- Domänenname
- Wert formatieren
- Text formatieren

Druckanzahl

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die aktuelle Druckanzahl auf dem Etikett aus.

Syntax

```
$Copies
```

Rückgabewert

Druckanzahl

If..Then..Else-Anweisung

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Fügt eine Wenn-Dann-Anweisung auf dem Etikett ein. Die Wenn-Dann-Anweisung dient dazu eine Bedingung auszuwerten und je nach Ergebnis weiterzuverfahren.

Syntax

```
$If (condition, thenValue, elseValue)
```

Parameter

condition
Ist die Bedingung gleich **true** oder **1** wird *value* zurückgegeben, ansonsten *elseValue*. Weitere Informationen finden Sie unter Mathematische Operatoren.

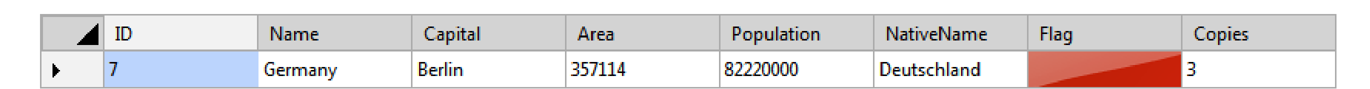
thenValue
Wert, der zurückgegeben wird wenn *condition* gleich **true** oder **1** ist.

elseValue
Wert, der zurückgegeben wird wenn *condition* gleich **false** oder **0** ist.

Rückgabewert

thenValue, wenn *condition* gleich **true** oder **1** ist, ansonsten *elseValue*.

Beispiele



ID	Name	Capital	Area	Population	NativeName	Flag	Copies
7	Germany	Berlin	357114	82220000	Deutschland		3

```
$If ($DbField ("Europe", "Area") <= 250000, "*", "**") -> "**"
```

Überprüfen, ob ein Datenbankfeld leer ist oder nicht

```
$If ($Length ($DBField (...)) == 0, "Das Datenbankfeld ist leer.", "Das Datenbankfeld ist nicht leer.")
```

Schichtdefinition

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die aktuelle Schichtbezeichnung auf dem Etikett aus.

Syntax

`$Shift`

Rückgabewert

Schichtbezeichnung oder druckerspezifische Variablendefinition.

Beispiele

Frühschicht -> 06:00 - 13:59
Spätschicht -> 14:00 - 21:59
Nachtschicht -> 22:00 - 05:59

Systemvariable (TrueType-Schrift)

`$Shift` -> "Frühschicht" (08:20)
`$Shift` -> "Spätschicht" (15:30)

Druckervariable (Druckerschrift)

`$Shift` -> "=SH()"

Siehe auch

- Schichtzeiten definieren

Schichtzeiten definieren

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Um die Schichtzeiten zu definieren, gehen Sie bitte folgendermaßen vor:

1. Klicken Sie in den **Etiketteneigenschaften** auf **Schichtzeiten definieren**.
   Das Dialogfeld **Schichtzeiten** wird geöffnet.
2. Klicken Sie auf:
   - , um eine neue Schicht zu definieren.
   - oder Entf, um die ausgewählte Schicht zu löschen.
   - oder doppelklicken Sie auf eine Schicht, um die Schichteinstellungen zu ändern.
   - oder Strg + ↑, um die ausgewählte Schicht einen Platz nach oben zu verschieben.
   - oder Strg + ↓, um die ausgewählte Schicht einen Platz nach unten zu verschieben.
3. Klicken Sie auf die Schaltfläche **OK**, um die geänderten Einstellungen zu speichern.

Hinweis

Bitte beachten Sie, dass sich die einzelnen Schichtzeiten nicht überlappen dürfen.

Falsch

Frühschicht -> 06:00 - 14:00
Spätschicht -> 14:00 - 22:00
Nachtschicht -> 22:00 - 06:00

Richtig

Frühschicht -> 06:00 - 13:59
Spätschicht -> 14:00 - 21:59
Nachtschicht -> 22:00 - 05:59

Etikettennummer

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die aktuelle Etikettennummer, innerhalb eines Druckauftrags, auf dem Etikett aus.

Syntax

```
$LabelNumber
```

Rückgabewert

Etikettennummer

Seitennummer

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt die aktuelle Seitennummer, innerhalb eines Druckauftrags, auf dem Etikett aus.

Syntax

```
$PageNumber
```

Rückgabewert

Seitennummer

Druckername

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den aktuellen Druckernamen auf dem Etikett aus.

Syntax

```
$PrinterName
```

Rückgabewert

Druckername

Benutzername

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den aktuellen Benutzernamen zurück.

Syntax

```
$UserName
```

Rückgabewert

Benutzername

Domänenname

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Gibt den Netzwerkdomänennamen des aktuellen Benutzers zurück.

Syntax

```
$UserDomainName
```

Rückgabewert

Netzwerkdomänenname

Wert formatieren

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Formatiert einen Wert.

Syntax

```
$Format (value, format, [Language=language, Type=type])
```

Parameter

value
Wert, der formatiert werden soll.

format
Gibt an, wie der Wert formatiert werden soll. Weitere Informationen finden Sie unter Formatzeichenfolgen.

language (optional, standarmäßig wird die unter Windows eingestellte Sprache verwendet)
Sprache, die zur Formatierung der Ausgabe verwendet werden soll. Weitere Informationen finden Sie unter Ländercodes.

type (optional)
Gibt den Typ von *value* an, der formatiert werden soll. Wird kein expliziter Typ angegeben wird zuerst auf Nummer, dann auf Datum/Uhrzeit und dann auf Text überprüft.
Number: *value* wird als Zahl interpretiert
Date/Time: *value* wird als Datum/Uhrzeit interpretiert
Text: *value* wird als Text interpretiert

Rückgabewert

Formatierter Text.

Beispiele

Zahl mit führenden Nullen darstellen

```
$Format (15, "00000") -> "00015"
$Format (-15, "00000") -> "-00015"
$Format (-15, "D5") -> "-00015"
```

Unterschiedliche Formatierung für negative Zahlen und Null

Hinweis: Sie können spezielle Formate für negative Zahlen und Null definieren. Verwenden Sie ein Semikolon ";" als Trennzeichen, um die Formatierung in zwei oder drei Abschnitte zu trennen. Der zweite Abschnitt ist für negative Zahlen, der dritte Abschnitt für Null.

```
$Format (15, "#;minus #") -> "15"
$Format (-15, "#;minus #") -> "minus 15"
$Format (0, "#;minus #;zero") -> "zero"
```

Unterschiedliche Sprachen verwenden

Hinweis: In den Beispielen wird, wenn keine Sprache angegeben ist, standardmäßig Deutsch (de-DE) verwendet.

```
$Format (-1234.56, "N2") -> "-1.234,56"
```

```
$Format (1234.56, "N2", Language="en-US") -> "-1,234.56"
$Format (1234.56, "N2", Language="fr-FR") -> "1 234,56"

$Format ($CurrentDateTime, "yyyy MMMM dddd") -> "2013 September Dienstag"
$Format ($CurrentDateTime, "yyyy MMMM dddd", Language="fr-FR") -> "2013 septembre mardi"
$Format ($CurrentDateTime, "yyyy MMMM dddd", Language="zh-CN") -> "2013 九月星期二"
```

Siehe auch

- Text formatieren

Fomatzeichenfolgen

Bei der Formatierung wird ein Wert mit Hilfe einer Formatzeichenfolge in die entsprechende Zeichenfolgendarstellung konvertiert. Eine Formatzeichenfolge ist eine Zeichenfolge, die einen oder mehrere vordefinierte Formatbezeichner enthält. Dies sind einzelne Zeichen oder Gruppen von Zeichen, die definieren, wie die Ausgabe formatiert werden soll.

Siehe auch

- Standardmäßige Zahlenformatzeichenfolgen
- Benutzerdefinierter Zahlenformatzeichenfolgen
- Standard-Formatzeichenfolgen für Datum und Uhrzeit
- Benutzerdefinierte Formatzeichenfolgen für Datum und Uhrzeit
- Textformatzeichenfolgen

Standardmäßige Zahlenformatzeichenfolgen

Eine Standardformatzeichenfolge für Zahlen hat die Form Axx, wobei A ein alphabetisches Zeichen (Formatbezeichner) und xx eine optionale Ganzzahl (Genauigkeitsangabe) ist. Die Genauigkeitsangaben reicht von 0 bis 99 und wirkt sich auf die Anzahl der Ziffern im Ergebnis aus. Jede Formatzeichenfolge, die mehr als ein alphabetisches Zeichen (einschließlich Leerzeichen) enthält, wird als benutzerdefinierte Zahlenformatzeichenfolge interpretiert. Weitere Informationen finden Sie unter Benutzerdefinierte Zahlenformatzeichenfolgen.

Die folgende Tabelle beschreibt die standardmäßigen Zahlenformatbezeichner.

Formatbezeichner	Name	Beschreibung	Beispiele
C oder c	Währung	Ergebnis: Währungswert. Genauigkeitsangabe: Anzahl der Dezimalstellen.	123.456 ("C", en-US) -> $123.46 123.456 ("C", fr-FR) -> 123,46 € 123.456 ("C", ja-JP) -> ¥123 -123.456 ("C3", en-US) -> ($123.456) -123.456 ("C3", fr-FR) -> -123,456 € -123.456 ("C3", ja-JP) -> -¥123.456
D oder d	Dezimal	Ergebnis: Ganzzahlige Ziffern mit optionalem Minuszeichen. Genauigkeitsangabe: Mindestanzahl von Ziffern.	1234 ("D") -> 1234 -1234 ("D6") -> -001234
E oder e	Exponential (wissenschaftlich)	Ergebnis: Exponentielle Notation. Genauigkeitsangabe: Anzahl der Dezimalstellen.	1052.0329112756 ("E", en-US) -> 1.052033E+003 1052.0329112756 ("e", fr-FR) -> 1,052033e+003 -1052.0329112756 ("e2", en-US) -> -1.05e+003 -1052.0329112756 ("E2", fr_FR) -> -1,05E+003
F oder f	Festkomma	Ergebnis: Ganze Zahlen und Dezimalzahlen mit optionalem Minuszeichen. Genauigkeitsangabe: Anzahl der Dezimalstellen.	1234.567 ("F", en-US) -> 1234.57 1234.567 ("F", de-DE) -> 1234,57 1234 ("F1", en-US) -> 1234.0 1234 ("F1", de-DE) -> 1234,0 -1234.56 ("F4", en-US) -> -1234.5600 -1234.56 ("F4", de-DE) -> -1234,5000
G oder g	Allgemein	Ergebnis: Die kompakteste Festkomma- oder wissenschaftliche Notation. Genauigkeitsangabe: Anzahl der signifikanten Stellen.	-123.456 ("G", en-US) -> -123.456 123.456 ("G", sv-SE) -> -123,456 123.4546 ("G4", en-US) -> 123.5 123.4546 ("G4", sv-SE) -> 123,5 -1.234567890e-25 ("G", en-US) -> -1.23456789E-25 -1.234567890e-25 ("G", sv-SE) -> -1,23456789E-25
N oder n	Zahl	Ergebnis: Ganze Zahlen und Dezimalzahlen, Gruppentrennzeichen und ein Dezimaltrennzeichen mit optionalem Minuszeichen. Genauigkeitsangabe: Anzahl der Dezimalstellen.	1234.567 ("N", en-US) -> 1,234.57 1234.567 ("N", ru-RU) -> 1 234,57 1234 ("N1", en-US) -> 1,234.0 1234 ("N1", ru-RU) -> 1 234,0 -1234.56 ("N3", en-US) -> -1,234.560 -1234.56 ("N3", ru-RU) -> -1 234,560
P oder p	Prozent	Ergebnis: Die Zahl multipliziert mit 100 und mit einem Prozentzeichen versehen. Genauigkeitsangabe: Anzahl der Dezimalstellen.	1 ("P", en-US) -> 100.00 % 1 ("P", fr-FR) -> 100,00 % -0.39678 ("P1", en-US) -> -39.7 % -0.39678 ("P1", fr-FR) -> -39,7 %
R oder r	Schleife	Ergebnis: Eine Zeichenfolge, die eine Schleife zu einem identischen Wert ausführen kann. Genauigkeitsangabe: Wird ignoriert.	123456789.12345678 ("R") -> 123456789.12345678 -1234567890.12345678 ("R") -> -1234567890.1234567

X oder x	Hexadezimal	Ergebnis: Eine Hexadezimalzeichenfolge. Genauigkeitsangabe: Anzahl von Ziffern in der Ergebniszeichenfolge.	255 ("X") -> FF -1 ("x") -> ff 255 ("x4") -> 00ff -1 ("X4") -> 00FF
Jedes andere Zeichen	Unbekannter Bezeichner		

Benutzerdefinierter Zahlenformatzeichenfolgen

Benutzerdefinierte Formatzeichenfolgen können aus einem oder mehreren Formatbezeichnern bestehen. Alle Zeichenfolgen, bei denen es sich nicht um standardmäßige Zahlenformatzeichenfolgen handelt, werden als benutzerdefinierte Formatzeichenfolgen für Zahlen interpretiert.

Die folgenden Tabelle beschreibt die benutzerdefinierten Zahlenformatbezeichner.

Formatbezeichner	Name	Beschreibung	Beispiele
0	0-Platzhalter	Ersetzt die Ziffer 0 ggf. durch eine entsprechende vorhandene Ziffer; andernfalls wird die Ziffer 0 in der Ergebniszeichenfolge angezeigt.	1234.5678 ("00000") -> 01235 0.45678 ("0.00", en-US) -> 0.46 0.45678 ("0.00", fr-FR) -> 0,46
#	Ziffernplatzhalter	Ersetzt das Nummernzeichen ggf. durch eine entsprechende vorhandene Ziffer; andernfalls wird keine Ziffer in der Ergebniszeichenfolge angezeigt.	1234.5678 ("#####") -> 1235 0.45678 ("#.##", en-US) -> .46 0.45678 ("#.##", fr-FR) -> ,46
.	Dezimaltrennzeichen	Bestimmt die Position des Dezimaltrennzeichens in der Ergebniszeichenfolge.	0.45678 ("0.00", en-US) -> 0.46 0.45678 ("0.00", fr-FR) -> 0,46
,	Gruppentrennzeichen und Zahlenskalierung	Das Zeichen "," wird sowohl als Bezeichner für Gruppentrennzeichen als auch als Bezeichner für Zahlenskalierung verwendet. Bei einer Verwendung als Bezeichner für Gruppentrennzeichen wird ein lokalisiertes Trennzeichen zwischen die einzelnen Gruppen eingefügt. Bei einer Verwendung als Bezeichner für Zahlenskalierung wird eine Zahl für jedes angegebene Zeichen durch 1000 geteilt.	Bezeichner für Gruppentrennzeichen: 2147483647 ("##,#", en-US) -> 2,147,483,647 2147483647 ("##,#", es-ES) -> 2.147.483.647 Bezeichner für Zahlenskalierung: 2147483647 ("#,#,,", en-US) -> 2,147 2147483647 ("#,#,,", es-ES) -> 2.147
%	Prozentplatzhalter	Multipliziert eine Zahl mit 100 und fügt ein lokalisiertes Prozentsymbol in die Ergebniszeichenfolge ein.	0.3697 ("%#0.00", en-US) -> %36.97 0.3697 ("%#0.00", el-GR) -> %36,97 0.3697 ("##.0 %", en-US) -> 37.0 % 0.3697 ("##.0 %", el-GR) -> 37,0 %
‰	Promilleplatzhalter	Multipliziert eine Zahl mit 1000 und fügt ein lokalisiertes Promillesymbol in die Ergebniszeichenfolge ein.	0.03697 ("#0.00‰", en-US) -> 36.97‰ 0.03697 ("#0.00‰", ru-RU) -> 36,97‰
\	Escapezeichen	Das Zeichen, das auf das Escapezeichen folgt, wird als Literal und nicht als benutzerdefinierter Formatbezeichner interpretiert.	987654 ("\###00\#") -> #987654#
;	Abschnittstrennzeichen	Definiert Abschnitte mit separaten Formatzeichenfolgen für positive und negative Zahlen sowie Nullen.	12.345 ("plus #0.0#;minus #0.0#;null") -> plus 12.35 0 ("plus #0.0#;minus #0.0#;null") -> null -12.345 ("plus #0.0#;minus #0.0#;null") -> minus 12.35
Jedes andere Zeichen		Das Zeichen wird unverändert in die Ergebniszeichenfolge kopiert.	68 ("# °") -> 68 °

Standard-Formatzeichenfolgen für Datum und Uhrzeit

Eine Standardformatzeichenfolge für Datums- und Uhrzeitwerte besteht aus einem alphabetischen Zeichen (Formatbezeichner). Jede Formatzeichenfolge, die mehr als ein alphabetisches Zeichen (einschließlich Leerzeichen) enthält, wird als benutzerdefinierte Datums- und Uhrzeitformatzeichenfolge interpretiert. Weitere Informationen finden Sie unter Benutzerdefinierte Datums- und Uhrzeitzeichenfolgen.

Die folgende Tabelle beschreibt die standardmäßigen Datums- und Uhrzeitformatbezeichner.

Formatbezeichner	Beschreibung	Beispiele
d	Kurzes Datum	15.06.2009 13:45:30 -> 6/15/2009 (en-US) 15.06.2009 13:45:30 -> 15/06/2009 (fr-FR) 15.06.2009 13:45:30 -> 2009/06/15 (ja-JP)
D	Langes Datum	15.06.2009 13:45:30 -> Monday, June 15, 2009 (en-US) 15.06.2009 13:45:30 -> 15 июня 2009 г.(ru-RU) 15.06.2009 13:45:30 -> Montag, 15.Juni 2009 (de-DE)
f	Vollständiges Datum (kurze Zeit)	15.06.2009 13:45:30 -> Monday, June 15, 2009 1:45 PM (en-US) 15.06.2009 13:45:30 -> Höhle 15 juni 2009 13:45 (sv-SE) 15.06.2009 13:45:30 -> Δευτέρα, 15 Ιουνίου 2009 1:45 μμ (el-GR)
F	Vollständiges Datum (lange Zeit)	15.06.2009 13:45:30 -> Monday, June 15, 2009 1:45:30 PM (en-US) 15.06.2009 13:45:30 -> den 15 juni 2009 13:45:30 (sv-SE) 15.06.2009 13:45:30 -> Δευτέρα, 15 Ιουνίου 2009 1:45:30 μμ (el-GR)
g	Allgemeines Datum (kurze Zeit)	15.06.2009 13:45:30 -> 6/15/2009 1:45 PM (en-US) 15.06.2009 13:45:30 -> 15/06/2009 13:45 (es-ES) 15.06.2009 13:45:30 -> 2009/6/15 13:45 (zh-CN)
G	Allgemeines Datum (lange Zeit)	15.06.2009 13:45:30 -> 6/15/2009 1:45:30 PM (en-US) 15.06.2009 13:45:30 -> 15/06/2009 13:45:30 (es-ES) 15.06.2009 13:45:30 -> 2009/6/15 13:45:30 (zh-CN)
M oder m	Tag/Monat	15.06.2009 13:45:30 -> June 15 (en-US) 15.06.2009 13:45:30 -> 15juni (da-DK) 15.06.2009 13:45:30 -> 15 Juni (id-ID)
R oder r	RFC1123	15.06.2009 13:45:30 - > Montag 15. Juni 2009 20:45:30 GMT
s	Sortierbares Datum	15.06.2009 13:45:30 -> 2009-06-15T13:45:30
t	Kurze Zeit	15.06.2009 13:45:30 -> 1:45 PM (en-US) 15.06.2009 13:45:30 -> 13:45 (hr-HR) 15.06.2009 13:45:30 -> 01:45 م (ar-EG)
T	Lange Zeit	15.06.2009 13:45:30 -> 1:45:30 PM (en-US) 15.06.2009 13:45:30 -> 13:45:30 (hr-HR) 15.06.2009 13:45:30 -> 01:45:30 م (ar-EG)
u	Universelles, sortierbares Datum	15.06.2009 13:45:30 -> 2009-06-15 20:45:30Z
U	Universelles Datum (Koordinierte Weltzeit)	15.06.2009 13:45:30 -> Monday, June 15, 2009 8:45:30 PM (en-US) 15.06.2009 13:45:30 -> den 15 juni 2009 20:45:30 (sv-SE) 15.06.2009 13:45:30 -> Δευτέρα, 15 Ιουνίου 2009 8:45:30 μμ (el-GR)
Y oder y	Jahr/Monat	15.06.2009 13:45:30 -> June, 2009 (en-US) 15.06.2009 13:45:30 -> juni 2009 (da-DK)

		15.06.2009 13:45:30 -> Juni 2009 (id-ID)

Benutzerdefinierte Formatzeichenfolgen für Datum und Uhrzeit

Benutzerdefinierte Formatzeichenfolgen können aus einem oder mehreren Formatbezeichnern bestehen. Alle Zeichenfolgen, bei denen es sich nicht um standardmäßige Datums- und Uhrzeitformatzeichenfolgen handelt, werden als benutzerdefinierte Formatzeichenfolgen für Datums- und Uhrzeitwerte interpretiert.

Die folgenden Tabelle beschreibt die benutzerdefinierten Datums- und Uhrzeitformatbezeichner.

Formatbezeichner	Beschreibung	Beispiele
d	Tag des Monats, von 1 bis 31	01.06.2009 13:45:30 -> 1 15.06.2009 13:45:30 -> 15
dd	Tag des Monats, von 01 bis 31	01.06.2009 13:45:30 -> 01 15.06.2009 13:45:30 -> 15
ddd	Abgekürzter Name des Wochentags	15.06.2009 13:45:30 -> Mon (en-US) 15.06.2009 13:45:30 -> Пн (ru-RU) 15.06.2009 13:45:30 -> lun. (fr-FR)
dddd	Vollständiger Name des Wochentags	15.06.2009 13:45:30 -> Monday (en-US) 15.06.2009 13:45:30 -> понедельник (ru-RU) 15.06.2009 13:45:30 -> lundi (fr-FR)
h	Stunde, von 1 bis 12 (12-Stunden-Format)	15.06.2009 01:45:30 -> 1 15.06.2009 13:45:30 -> 1
hh	Stunde, von 01 bis 12 (12-Stunden-Format)	15.06.2009 01:45:30 -> 01 15.06.2009 13:45:30 -> 01
H	Stunde, von 0 bis 23 (24-Stunden-Format)	15.06.2009 01:45:30 -> 1 15.06.2009 13:45:30 -> 13
HH	Stunde, von 00 bis 23 (24-Stunden-Format)	15.06.2009 01:45:30 -> 01 15.06.2009 13:45:30 -> 13
m	Minute, von 0 bis 59	15.06.2009 01:09:30 -> 9 15.06.2009 13:09:30 -> 9
mm	Minute, von 00 bis 59	15.06.2009 01:09:30 -> 09 15.06.2009 13:09:30 -> 09
M	Monat, von 1 bis 12	15.06.2009 13:45:30 -> 6
MM	Monat, von 01 bis 12	15.06.2009 13:45:30 -> 06
MMM	Abgekürzter Name des Monats	15.06.2009 13:45:30 -> Jun (en-US) 15.06.2009 13:45:30 -> juin (fr-FR) 15.06.2009 13:45:30 -> Jun (zu-ZA)
MMMM	Vollständiger Name des Monats	15.06.2009 13:45:30 -> June (en-US) 15.06.2009 13:45:30 -> juni (da-DK) 15.06.2009 13:45:30 -> uJuni (zu-ZA)
s	Sekunde, von 0 bis 59	15.06.2009 13:45:09 -> 9
ss	Sekunde, von 00 bis 59	15.06.2009 13:45:09 -> 09
y	Jahr, von 0 bis 99	01.01.0001 00:00:00 -> 1 01.01.0900 00:00:00 -> 0 01.01.1900 00:00:00 -> 0 15.06.2009 13:45:30 -> 9
yy	Jahr, von 00 bis 99	01.01.0001 00:00:00 -> 01 01.01.0900 00:00:00 -> 00 01.01.1900 00:00:00 -> 00 15.06.2009 13:45:30 -> 09
yyy	Jahr, mit einem Minimum von drei Ziffern	01.01.0001 00:00:00 -> 001 01.01.0900 00:00:00 -> 900 01.01.1900 00:00:00 -> 1900 15.06.2009 13:45:30 -> 2009
yyyy	Jahr (vierstellig)	01.01.0001 00:00:00 -> 0001

		01.01.0900 00:00:00 -> 0900 01.01.1900 00:00:00 -> 1900 15.06.2009 13:45:30 -> 2009
yyyyy	Jahr (fünfstellig)	01.01.0001 00:00:00 -> 00001 15.06.2009 13:45:30 -> 02009
:	Zeittrennzeichen	15.06.2009 13:45:30 -> : (en-US) 15.06.2009 13:45:30 -> .(it-IT) 15.06.2009 13:45:30 -> : (ja-JP)
/	Datumstrennzeichen	15.06.2009 13:45:30 -> / (en-US) 15.06.2009 13:45:30 -> - (ar-DZ) 15.06.2009 13:45:30 -> .(tr-TR)
\	Escapezeichen	15.06.2009 13:45:30 (h \h) -> 1 h
Jedes andere Zeichen	Das Zeichen wird unverändert in die Ergebniszeichenfolge kopiert	15.06.2009 01:45:30 (arr hh:mm t) -> arr 01:45 A

Textformatzeichenfolgen

Die Formatzeichenfolge wird verwendet um die Textdarstellung zu definieren.

Die folgende Tabelle beschreibt die Textformatbezeichner.

Formatbezeichner	Name	Beschreibung	Beispiele
0	Ziffernplatzhalter	Ersetzt den Platzhalter durch eine entsprechende vorhandene Ziffer (0-9); andernfalls wird die Ziffer 0 in der Ergebniszeichenfolge angezeigt.	abc-123-DEF ("00000") -> 12300 abc-123-DEF ("!00000") -> 00123
9	Ziffernplatzhalter	Ersetzt den Platzhalter durch eine entsprechende vorhandene Ziffer (0-9); andernfalls wird ein Leerzeichen in der Ergebniszeichenfolge angezeigt.	abc-123-DEF ("!99999") -> 123
#	Ziffernplatzhalter (optional)	Ersetzt den Platzhalter durch eine entsprechende vorhandene Ziffer (0-9) oder ein entsprechend vorhandenes Plus- oder Minuszeichen; andernfalls wird keine Ziffer in der Ergebniszeichenfolge angezeigt.	abc-123-DEF (#####) -> -123
L	Buchstabenplatzhalter	Ersetzt den Platzhalter durch einen entsprechend vorhandenen Buchstaben (a-Z); andernfalls wird ein Leerzeichen in der Ergebniszeichenfolge angezeigt.	abc-123-DEF (LLLLL) -> abc12
?	Buchstabenplatzhalter (optional)	Ersetzt den Platzhalter durch einen entsprechend vorhandenen Buchstaben (a-Z); andernfalls wird kein Buchstabe in der Ergebniszeichenfolge angezeigt.	
&	Zeichenplatzhalter	Ersetzt den Platzhalter durch ein entsprechend vorhandenes beliebiges Zeichen; andernfalls wird ein Leerzeichen in der Ergebniszeichenfolge angezeigt.	abc-123-DEF (&&&&&) -> abc-1
C	Zeichenplatzhalter (optional)	Ersetzt den Platzhalter durch ein entsprechend vorhandenes beliebiges Zeichen; andernfalls wird kein Zeichen in der Ergebniszeichenfolge angezeigt.	
A	Alphanumerischer Platzhalter	Ersetzt den Platzhalter durch ein entsprechend vorhandenes alphanumerisches Zeichen (0-9, a-Z); andernfalls wird ein Leerzeichen in der Ergebniszeichenfolge angezeigt.	abc-123-DEF (AAAAA) -> abc12
a	Alphanumerischer Platzhalter (optional)	Ersetzt den Platzhalter durch ein entsprechend vorhandenes alphanumerisches Zeichen (0-9, a-Z); andernfalls wird kein Zeichen in der Ergebniszeichenfolge angezeigt.	
<	In Kleinbuchstaben umwandeln	Konvertiert alle nachfolgenden Zeichen (a-Z) in Kleinbuchstaben.	abcDEF (<LLLLLL) -> abcdef
>	In Großbuchstaben umwandeln	Konvertiert alle nachfolgenden Zeichen (a-Z) in Großbuchstaben.	abcDEF (>LLLLLL) -> ABCDEF
|	Umwandlung deaktivieren	Deaktiviert eine vorangegangene Umwandlung in Klein- oder Großbuchstaben.	abcDEF (>LL<LL|LL) -> ABcdEF

!	Rechtsbündige Ausgabe	Gibt die Ergebniszeichenfolge rechtsbündig aus.	123 (!00000) -> 00123
^	Rechtsbündige Ausgabe deaktivieren	Deaktiviert eine vorangegangene rechstbündige Ausgabe.	
\	Escapezeichen	Das Zeichen, das auf das Escapezeichen folgt, wird als Literal und nicht als benutzerdefinierter Formatbezeichner interpretiert.	abcDEF (LLL\LLLL) -> abcLDEF
Jedes andere Zeichen	Literale		abc (L-L-L) -> a-b-c abc (>L:L:L) -> A:B:C

Ländercodes

Ländercode	Sprache	
zh-CN	Chinesisch (Volksrepublik China)	中文(中华人民共和国)
zh-Hans	Chinese (vereinfacht)	中文(简体)
da	Dänisch	dansk
de	Deutsch	Deutsch
de-DE	Deutsch (Deutschland)	Deutsch (Deutschland)
de-LI	Deutsch (Liechtenstein)	Deutsch (Liechtenstein)
de-LU	Deutsch (Luxemburg)	Deutsch (Luxemburg)
de-CH	Deutsch (Schweiz)	Deutsch (Schweiz)
en	Englisch	English
en-GB	English (Großbritannien)	English (United Kingdom)
en-US	English (Vereinigte Staaten)	English (United States)
fi	Finnisch	suomi
fr	Französisch	français
fr-BE	Französisch (Belgien)	français (Belgique)
fr-FR	Französisch (Frankreich)	français (France)
fr-LU	Französisch (Luxemburg)	français (Luxembourg)
fr-CH	Französisch (Schweiz)	français (Suisse)
el	Griechisch	ελληνικά
he	Hebräisch	עברית
nl	Holländisch	Nederlands
nl-BE	Holländisch (Belgien)	Nederlands (België)
nl-NL	Holländisch (Niederlande)	Nederlands (Nederland)
it	Italienisch	italiano
it-IT	Italienisch (Italien)	italiano (Italia)
it-CH	Italienisch (Schweiz)	italiano (Svizzera)
ja	Japanisch	日本語
pl	Polnisch	polski
pt-BR	Portugiesisch (Brasilien)	Português (Brasil)
pt-PT	Portugiesisch (Portugal)	Português (Portugal)
ru	Russisch	русский
es	Spanisch	español
es-ES	Spanisch (Spanien)	español (España)
cs	Tschechisch	čeština
tr	Türkisch	Türkçe
hu	Ungarisch	magyar

Eine vollständige Liste aller Ländercodes finden Sie hier.

Text formatieren

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Formatiert einen Text.

Syntax

```
$FormatText (value, format)
```

Parameter

value
Text, der formatiert werden soll.

format
Gibt an, wie der Text formatiert werden soll. Weitere Informationen finden Sie unter Textformatzeichenfolgen.

Rückgabewert

Formatierter Text.

Beispiele

```
$FormatText ("123456", "####,##") -> "1234,56"
$FormatText ("1234", "0000-00") -> 0012-34
```

Siehe auch

- Wert formatieren

Druckervariablen

Mit Hilfe dieser Variablen können druckerinterne Variablen auf dem Etikett definiert werden. Druckerinterne Variablen werden, im Gegensatz zu Systemvariablen, während des Druckauftrags vom Drucker verwaltet und berechnet.

Hinweis

- Druckervariablen können nur in Textfeldern mit Druckerschriften und Barcodes, die nicht grafisch übertragen werden, verwendet werden.
- Je Text- oder Barcodefeld kann immer nur **eine** Druckervariable definiert werden.

Unterstützte Druckervariablen

- Datum/Uhrzeit (Drucker)
- Kettenfeld (Drucker)
- Benutzereingabe (Drucker)
- Numerator (Drucker)
- Erweiterter Numerator (Drucker)
- Schichtdefinition
- Prüfziffer (Drucker)

Datum/Uhrzeit (Drucker)

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert eine druckerinterne Datums- und Uhrzeitvariable.

Syntax

```
$PrnDateTime (format, [UpdateInterval=updateInterval, MonthOffset=monthOffset, DayOffset=dayOffset,
MinOffset=minOffset, CorrectMonth=correctMonth])
```

Parameter

format
Gibt an, wie der Wert formatiert werden soll. Weitere Informationen finden Sie unter Druckerspezifische Datums- und Uhrzeitformatzeichenfolgen.

updateInterval (optional, Standard = 0)
Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
0: Am Druckbeginn
1: Nach jedem Etikett

monthOffset (optional, Standard = 0)
Monatsoffset (wird zum aktuellen Datum hinzugezählt)

dayOffset (optional, Standard = 0)
Tagesoffset (wird zum aktuellen Datum hinzugezählt)

minOffset (optional, Standard = 0)
Minutenoffset (wird zur aktuellen Uhrzeit hinzugezählt)

correctMonth (optional, Standard = false)
Monatskorrektur
false oder 0: In den nächsten Monat wechseln
true oder 1: Aktuellen Monat beibehalten

Rückgabewert

Druckerspezifische Variablendefinition.

Beispiele

Aktuelle Druckereinstellung: 25.02.2014 14:21:25

```
$PrnDateTime ("DD.MO.YYYY") -> "=CL(0;0;0;0;0)<DD.MO.YYYY>" -> 25.02.2014
$PrnDateTime ("HH:MI:SS", UpdateInterval=1, MinOffset=-60) -> "=CL(0;0;1;-60;0)<HH:MI:SS>" -> 13:21:25
```

Siehe auch

- Datum/Uhrzeit (System)

Druckspezifische Datums- und Uhrzeitformatzeichenfolgen

Die Formatzeichenfolge wird verwendet um die Textdarstellung einer druckerinternen Datums- und Uhrzeitvariable zu definieren.

Die folgenden Tabelle beschreibt die Datums- und Uhrzeitformatbezeichner.

Formatbezeichner	Beschreibung	Beispiele
HH	Stunde, von 00 bis 23 (24-Stunden-Format)	15.06.2009 01:45:30 -> 01 15.06.2009 13:45:30 -> 13
HE	Stunde, von 00 bis 23 (12-Stunden-Format)	15.06.2009 01:45:30 -> 01 15.06.2009 13:45:30 -> 01
MI	Minute, von 00 bis 59	15.06.2009 01:09:30 -> 09 15.06.2009 13:09:30 -> 09
SS	Sekunde, von 00 bis 59	15.06.2009 13:45:09 -> 09
AM, Am oder am	AM/PM Ausgabe	15.06.2009 13:45:09 -> PM (AM) 15.06.2009 13:45:09 -> p.m. (Am) 15.06.2009 13:45:09 -> pm (am)
DD	Tag des Monats, von 01 bis 31	01.06.2009 13:45:30 -> 01 15.06.2009 13:45:30 -> 15
MO	Monat, von 01 bis 12	15.06.2009 13:45:30 -> 06
YYYY	Jahr (vierstellig)	15.06.2009 13:45:30 -> 2009
YY	Jahr, von 00 bis 99	15.06.2009 13:45:30 -> 09
Y	Jahr, von 0 bis 9	15.06.2009 13:45:30 -> 9
WW	Kalenderwoche	15.06.2009 13:45:30 -> 25
DW	Wochentag, von 0 (Sonntag) bis 6 (Samstag)	15.06.2009 13:45:30 -> 1
DW1	Wochentag, von 1 (Sonntag) bis 7 (Samstag)	15.06.2009 13:45:30 -> 2
Dwx	Wochentag Für x kann ein beliebiges ASCII-Zeichen eingesetzt werden, von dem ab, beginnend mit Sonntag, fortlaufend weitergezählt wird.	
DOWxxxxxxx	Wochentag (variable) Für x kann ein beliebieges ASCII-Zeichen eingesetzt werden. Das erste 'x' steht für Sonntag, das nächste für Montag, usw. bis Samstag. **Hinweis:** Für jeden Wochentag muss ein Zeichen angegeben werden.	
DOY	Tag im Jahr, von 001 bis 365	15.06.2009 13:45:30 -> 166
DY	Tag im Jahr, von 000 bis 364	15.06.2009 13:45:30 -> 165
\	Escapezeichen	
Jedes andere Zeichen	Das Zeichen wird unverändert in die Ergebniszeichenfolge kopiert	

Die folgende Tabelle beschreibt die länderspezifischen Datums- und Uhrzeitformate.

Formatbezeichner	Beschreibung	Beispiele
xMO	Abgekürzter Name des Monats	15.06.2009 13:45:30 -> JN (CMO) 15.06.2009 13:45:30 -> JUN (DMO) 15.06.2009 13:45:30 -> GUI (IMO)
xSO	Vollständiger Name des Monats	15.06.2009 13:45:30 -> June (ESO) 15.06.2009 13:45:30 -> Juin (FSO) 15.06.2009 13:45:30 -> Junio (SSO)
xSD	Abgekürzter Name des Wochentags	15.06.2009 13:45:30 -> MO (GSD)

		15.06.2009 13:45:30 -> MA (NSD) 15.06.2009 13:45:30 -> LUN (SSD)
xLD	Vollständiger Name des Wochentags	15.06.2009 13:45:30 -> Monday (ELD) 15.06.2009 13:45:30 -> Montag (GLD) 15.06.2009 13:45:30 -> Mandag (OLD)

Für x kann die Länderkennung der gewünschten Sprache eingesetzt werden.

C = Kanadisch
D = Dänisch
E = Englisch
F = Französisch
G = Deutsch
I = Italienisch
N = Niederländisch
O = Norwegisch
S = Spanisch
U = Finnisch
W = Schwedisch

Kettenfeld (Drucker)

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert ein druckerinternes Kettenfeld.

Syntax

```
$PrnFieldLink (value, [value, ...])
```

Parameter

value

Bezeichnung der Kettenelemente (Feldname oder Textkonstante). Eine Textkonstante muss in Anführungsstriche gesetzt werden. Die Anführungszeichen werden nicht gedruckt.
Hinweis: Für die Verkettung dürfen nur druckerinterne Felder verwendet werden.

Rückgabeparameter

Druckerspezifische Variablendefinition.

Beispiele

`$PrnFieldLink (ID01, "Textkonstante", ID02)` -> "SC=(0;"Textkonstante";1)"

Siehe auch

- Feldinhalt auslesen

Benutzereingabe (Drucker)

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert eine druckerinterne Benutzereingabe.

Syntax

```
$PrnUserInput (prompt, text, [StartPos=startPos, AllowableChars=allowableChars,
SkipSpecialChars=skipSpecialChars, PrintAlignment=printAlignment, InputAlignment=inputAlignment])
```

Parameter

prompt
Der Eingabeaufforderungstext, der in der ersten Zeile im Druckerdisplays angezeigt wird.

text
Der Eingabetext, der in der zweiten Zeile im Druckerdisplay angezeigt wird.

startPos (optional, Standard = 0)
Startposition für die Eingabe. Ist die Startposition gleich 0, so wird die Anzahl von Zeichen in *text* an den Drucker übertragen.

allowableChars (optional, Standard = 0)
Gibt an, welche Zeichen für die Eingabe zulässig sind.
0: Nummerisch
1: Alphanummerisch

skipSpecialChars (optional, Standard = 0)
Gibt an, ob Sonderzeichen bei der Eingabe beibehalten werden sollen oder nicht.
0: Sonderzeichen nicht überspringen
1: Sonderzeichen überspringen

printAlignment (optional, Standard = 0)
Druckausrichtung
0: Rechtsbündig
1: Linksbündig

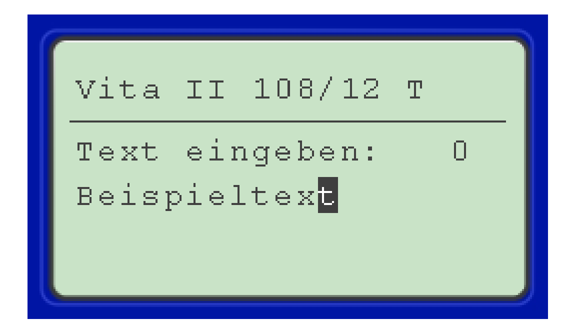
inputAlignment (optional, Standard = 0)
Eingabeausrichtung
0: Rechtsbündig
1: Linksbündig

Rückgabewert

Druckerspezifische Variablendefinition.

Beispiele

`$PrnUserInput ("Text eingeben:", "Beispieltext", StartPos=0, AllowableChars=1)` -> "=UG(12;1;0;0;0;"Text eingeben:")<Beispieltext>"



Siehe auch

- Benutzereingabe (System)

Numerator (Drucker)

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert einen druckerinternen Numerator.

Syntax

```
$PrnCounter (value, [Prompt=prompt, UpdateInterval=updateInterval, Increment=increment, Pos=pos,
Radix=radix, Mode=mode, ResetTime=resetTime, ResetValue=resetValue])
```

Parameter

value

Aktueller Startwert.
Hinweis: Die Anzahl der Stellen legt das Ausgabeformat fest (maximal "999999999").

prompt (optional, Standard = leer)

Ist ein Eingabeaufforderungstext definiert, wird der Startwert am Druckbeginn abgefragt.

updateInterval (optional, Standard = 1)

Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
1: Nach jedem Etikett
n: Nach n Etiketten

increment (optional, Standard = 1)

Schrittweite.

pos (optional, Standard = 0)

Definiert die Stelle, an der der Numerator zu zählen beginnt. Ist die Position gleich 0, so wird die Anzahl von Zeichen in *value* an den Drucker übertragen.

radix (optional, Standard = 10)

Radix, Basis des Numerators (1-36)
1: Alphabetisch (A-Z)
2: Binär (0, 1)
8: Oktal (0-7)
10: Dezimal (0-9)
16: Hexadezimal (0-9, A-F)
36: Alphanumerisch (0-9, A-Z)

mode (optional, Standard = 1)

Betriebsart
0: Startwert manuell zurücksetzen
1: Startwert manuell zurücksetzen (automatischer Überlauf)
2: Startwert am Drucker eingeben
3: Startwert (= letzter Endwert) am Drucker eingeben
4: Startwert am Zyklusende zurücksetzen
5: Startwert über I/O-Signal zurücksetzen
6: Startwert zeitgesteuert zurücksetzen
7: Startwert zeitgesteuert zurücksetzen (Startwert am Drucker eingeben)

resetTime (optional, nur für Betriebsart 6 und 7)

Uhrzeit, an der der Startwert zurückgesetzt werden soll.
Format: "HH:MM"

resetValue (optional, nur für Betriebsart 6 und 7)
Wert, auf den der Startwert zurückgesetzt werden soll. Wird kein Wert angegeben, so wird der Numerator auf den ursprünglichen Startwert zurückgesetzt.

Rückgabewert

Druckerspezifische Variablendefinition.

Beispiele

`$PrnCounter ("0001", Mode=1)` -> "=CN(10;1;4;+1;1)0001"
`$PrnCounter ("1234", Mode=7, ResetTime="06:00", ResetValue="0001")` -> "=CN(10;7;4;+1;1;06:00;0001)1234"

Siehe auch

- Erweiterter Numerator (Drucker)
- Numerator (System)

Erweiterter Numerator (Drucker)

Benötigte Programmvariante	**BASIC, PROFESSIONAL**
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert einen druckerinternen Numerator.

Syntax

```
$PrnCounterExt (value, [Prompt=prompt, UpdateInterval=updateInterval, Increment=increment,
MinValue=minValue, MaxValue=maxValue, TrimLeft=trimLeft, Mode=mode])
```

Parameter

value

Aktueller Startwert.
Hinweis: Die Anzahl der Stellen legt das Ausgabeformat fest (maximal "999999999").

prompt (optional, Standard = leer)

Ist ein Eingabeaufforderungstext definiert, wird der Startwert am Druckbeginn abgefragt.

updateInterval (optional, Standard = 1)

Gibt an, wie oft die Variable während eines Druckauftrages upgedatet werden soll.
1: Nach jedem Etikett
n: Nach n Etiketten

increment (optional, Standard = 1)

Schrittweite.

minValue (optional, Standard = 0)

Minimaler Wert.

maxValue (optional, Standard = leer)

Maximaler Wert. Wird kein *maxValue* angegeben wird standardmäßig die Stellenanzahl des Startwertes verwendet um einen Maximalwert zu berechnen.

Startwert	Berechneter Maximalwert
0001	9999
01	99

trimLeft (optional, Standard = false)

true oder 1: Führende Nullen bei der Ausgabe unterdücken
false oder 0: Führnde Nullen bei der Ausgabe anzeigen

mode (optional, Standard = 5)

Betriebsart
0: Startwert manuell zurücksetzen
1: Startwert manuell zurücksetzen (automatischer Überlauf)
2: Startwert am Drucker eingeben
3: Startwert (= letzter Endwert) am Drucker eingeben
4: Startwert am Zyklusende zurücksetzen
5: Startwert manuell zurücksetzen (auf Min/Max)
6: Startwert manuell zurücksetzen (auf Startwert)
7: Startwert manuell zurücksetzen (Druck anhalten)

Rückgabewert

Druckerspezifische Variablendefinition.

Beispiele

`$PrnCounterExt ("0050", Increment=1, UpdateInterval=1, MinValue=1, MaxValue=999)` ->
"=CC(+1,1,5,0,1,999)0050" -> 50, 51, ... 999, 1, 2, ...

Siehe auch

- Numerator (Drucker)
- Numerator (System)

Prüfziffer (Drucker)

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Definiert eine druckerinterne Prüfziffernberechnung.

Syntax

```
$PrnCheckDigit (data, checkDigitMethod)
```

Parameter

data

Nutzzeichenfolge (Feldname oder Textkonstante), für die die Prüfsumme berechnet werden soll. Eine Textkonstante muss in Anführungsstriche gesetzt werden. Die Anführungszeichen werden nicht gedruckt.
Hinweis: Für die Verkettung dürfen nur druckerinterne Felder verwendet werden.

checkDigitMethod

Methode, nach der die Prüfziffer berechnet werden soll.

Methode	Beschreibung
MOD10	Modulo 10
MOD11	Modulo 11
MOD43	Modulo 43
MOD47_15	Modulo 47 (Gewichtung 15)
MOD47_20	Modulo 47 (Gewichtung 20)
MOD103	Modulo 103

Rückgabewert

Druckerspezifische Variablendefinition.

Beispiele

```
$PrnCheckDigit (ID01, MOD10) -> "CD=(0;0;0)"
$PrnCheckDigit ("123456789012", MOD10) -> "=CD("123456789012";0;0)"
```

Siehe auch

- Prüfziffer (System)

Barcodes

Die folgende Tabelle enthält die unterstützten Barcodetypen.

Code	Beispiel	Beschreibung
Aztec Code		2D Barcode, entwickelt von der Firma Welch Allyn.
Aztec Runes		2D Barcode auf Basis des Aztec Codes.
Codabar	123456	Numerischer Barcode, dessen Zeichenvorrat aus Ziffern und Sonderzeichen besteht.
Codablock F		2D Barcode auf Basis des Code 128.
Code 128	ABCabc	Alphanumerischer Barcode, dessen Zeichenvorrat den gesamten ASCII-Zeichensatz umfasst.
Code 128 (Zeichensatz A)	ABCDEF	Alphanumerischer Barcode, dessen Zeichenvorrat aus Ziffern, Großbuchstaben und Sonderzeichen besteht.
Code 128 (Zeichensatz B)	ABCabc	Alphanumerischer Barcode, dessen Zeichenvorrat aus Ziffern, Buchstaben und Sonderzeichen besteht.
Code 2/5 Industrial	123456	Numerischer Barcode

Code 2/5 Interleaved	123456	Numerischer Barcode mit gerader Stellenanzahl.
Code 39	*ABCDEF*	Alphanumerischer Barcode, dessen Zeichenvorrat aus Ziffern, Großbuchstaben, Sonderzeichen und Leerzeichen besteht.
Code 39 (Full ASCII)	*ABCabc*	Alphanumerischer Barcode auf Basis des Code 39, dessen Zeichenvorrat den gesamten ASCII-Zeichensatz umfasst.
Code 93	ABCDEF	Alphanumerischer Barcode, dessen Zeichenvorrat aus Ziffern, Großbuchstaben, Sonderzeichen und Leerzeichen besteht.
Code 93 (Full ASCII)	ABCabc	Alphanumerischer Barcode auf Basis des Code 93, dessen Zeichenvorrat den gesamten ASCII-Zeichensatz umfasst. **Hinweis:** Dieser Barcode wird grafisch übertragen.
DataMatrix		2D Barcode, entwickelt von der Firma Acuity Corp.
Deutsche Post Identcode	01.234 567.890 5	Numerischer Barcode auf Basis des Code 2/5 Interleaved mit geänderter Prüfziffernberechnung.
Deutsche Post Leitcode	01234.567.890.12 0	Numerischer Barcode auf Basis des Code 2/5 Interleaved mit geänderter Prüfziffernberechnung.
EAN-13, GTIN-13	1 234567 890128	Numerischer Barcode.
EAN-13 + 2 Stellen	1 234567 890128 12	EAN 13 mit zweistelligem Zusatzsymbol.

EAN-13 + 5 Stellen	1 234567 890128 12345	EAN 13 mit fünfstelligem Zusatzsymbol.
EAN-8, GTIN-8	1234 5670	Numerischer Barcode.
GS1 DataBar	(01)00614141999996	Alphanumerischer Barcode.
GS1 DataMatrix		2D Barcode.
GS1-128	(01)00614141999996	Alphanumerischer Barcode.
ITF-14, SCC-14	00614141999996	Numerischer Barcode auf Basis des Code 2/5 Interleaved.
MaxiCode		2D Barcode.
PDF417		2D Barcode.
Pharmacode	123456	Numerischer Barcode.
PZN	PZN - 12345684	Numerischer Barcode auf Basis des Code 39.

QR Code		2D Barcode.
UPC-A, GTIN-12	1 23456 78901 2	Numerischer Barcode.
UPC-E	0 123456 5	Numerischer Barcode.

1D Barcodes

Lineare Barcodes bestehen aus einer Zeile und kodieren die Informationen in Form von Strichen.

Unterstützte Barcodes

- Codabar
- Code 128
  - Code 128 (Zeichensatz A)
  - Code 128 (Zeichensatz B)
- Code 2/5 Industrial
- Code 2/5 Interleaved
- Code 39
- Code 39 (Full ASCII)
- Code 93
- Code 93 (Full ASCII)
- Deutsche Post Identcode
- Deutsche Post Leitcode
- EAN-13
  - EAN-13 + 2 Stellen
  - EAN-13 + 5 Stellen
- EAN-8
- GTIN-8
- GTIN-12
- GTIN-13
- ITF-14
- Pharmacode
- PZN
- SCC-14
- UPC-A
- UPC-E

Codabar



Der **Codabar** wird hauptsächlich in Bibliotheken, in der Fotobranche und medizinischen Bereichen (Blutbanken) verwendet. Der **Codabar** ist ein universeller, numerischer Barcode der zusätzlich zu den Ziffern 0-9 noch 6 Sonderzeichen enthält. Die Anzahl der darstellbaren Zeichen ist vom Code nicht vorgegeben.

Zusätzlich sind vier verschiedene Start-/Stoppzeichen (A-D) definiert, d.h. jeder Code muss mit A, B, C oder D beginnen und enden. Die Start-/Stoppzeichen dürfen jedoch im Barcode selbst nicht verwendet werden.

Jedes Zeichen des Codes besteht aus elf Modulen, vier Striche und drei Lücken. Eine vierte Lücke ist immer schmal.

Länge	Variabel
Zeichensatz	Ziffern 0-9 Sonderzeichen - $: / . +
Prüfziffer	Optional Modulo 16

Code 128



Der **Code 128** ist ein universeller, alphanumerischer Barcode der hauptsächlich im Speditions-/Transportgewerbe, auf Ausweisen und in der Lagerhaltung/Distribution eingesetzt wird.

Der **Code 128** kann alle 128 ASCII Zeichen darstellen. Die Verwendung der Prüfziffer ist für den **Code 128** vorgeschrieben. Durch die Nutzung vier verschiedener Breiten für Striche und Balken ist die Informationsdichte sehr hoch.

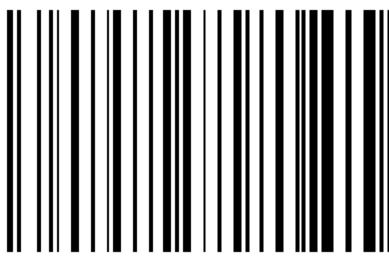
Der Aufbau eines **Code 128** Barcodesymbols besteht aus einem Startzeichen, Nutzdaten, Prüfziffer und dem Stoppzeichen. Vor dem Startzeichen und hinter dem Stoppzeichen muss eine Ruhezone mit einer Breite von mindestens 10 Modulen definiert sein.

Länge	Variabel
Zeichensatz	ASCII
Prüfziffer	Modulo 103

Siehe auch

- Code 128 (Zeichensatz A)
- Code 128 (Zeichensatz B)
- GS1-128

Code 128 (Zeichensatz A)



Sonderfall des Code 128.

Länge	Variabel
Zeichensatz	Ziffern 0-9 Großbuchstaben A-Z Sonderzeichen
Prüfziffer	Modulo 103

Siehe auch

- Code 128
- Code 128 (Zeichensatz B)

Code 128 (Zeichensatz B)



Sonderfall des Code 128.

Länge	Variabel
Zeichensatz	Ziffern 0-9 Groß- und Kleinbuchstaben A-z
Prüfziffer	Modulo 103

Siehe auch

- Code 128
- Code 128 (Zeichensatz A)

Code 2/5 Industrial



Der **Code 2/5 Industrial** ist ein numerischer Barcode der die Ziffern 0-9 beinhaltet. Der Code wird hauptsächlich in der Industrie und vor allem in der Transport-/ Lagertechnik angewandt. Die Verwendung der Prüfziffer ist nicht vorgeschrieben.

Da die Informationsdichte des Codes gering ist und sein Platzverbrauch sehr hoch ist wird er heutzutage kaum noch verwendet.

Der Barcode hat seinen Namen weil jede Ziffer in 5 Strichen kodiert ist, davon zwei breite Striche und drei schmale. Die Lücken zwischen den Strichen enthalten keinerlei Informationen.

Länge	Variabel
Zeichensatz	Ziffern 0-9
Prüfziffer	Optional Modulo 10 Modulo 10 (Luhn-Algorithmus)

Siehe auch

- Code 2/5 Interleaved

Code 2/5 Interleaved



Der **Code 2/5 Interleaved** ist eine Variante des Code 2/5 Industrial, mit dem Unterschied, dass eine Ziffer an ungerader Position (mit Strichen dargestellt), mit einer Ziffer an gerader Position (mit Lücken dargestellt), verschachtelt wird. Dadurch ergibt sich eine höhere Informationsdichte.

Zur Darstellung einer ungeraden Anzahl von Ziffern muss eine Null vorangestellt oder eine Prüfziffer angehängt werden.

Der Code wird hauptsächlich in der Industrie, vor allem im Logistikbereich eingesetzt.

Länge	Variabel (gerade Stellenanzahl)
Zeichensatz	Ziffern 0-9
Prüfziffer	Optional Modulo 10 Modulo 10 (Luhn-Algorithmus)

Siehe auch

- Code 2/5 Industrial

Code 39

ABCDEF

Der **Code 39** ist ein alphanumerischer Barcode der hauptsächlich bei Paketdiensten, in der Elektronik- und Chemieindustrie, im Gesundheitssektor und bei Speditionen eingesetzt wird.

Jedes Zeichen des Codes besteht aus neun Modulen, fünf Striche und vier Lücken. Drei von den Elementen sind breit und sechs schmal. Dadurch ist eine Selbstprüfung des Barcodes möglich.

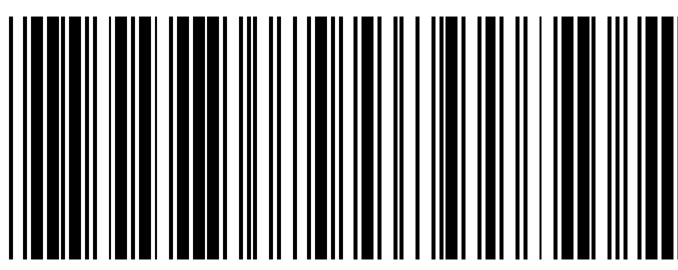
Der **Code 39** dominierte früher die Anwendungen die einen alphanumerischen Barcode zur Kodierung forderten. Aufgrund seiner niedrigen Informationsdichte und der geringen Zeichensatzauswahl wird er aktuell häufig durch den Code 128 ersetzt.

Länge	Variabel
Zeichensatz	Ziffern 0-9 Großbuchstaben A-Z Sonderzeichen - . $ / + % Leerzeichen
Prüfziffer	Optional Modulo 43 Modulo 11 (Gewichtung 7) Modulo 10 (Luhn-Algorithmus)

Siehe auch

- Code 39 (Full ASCII)

Code 39 (Full ASCII)



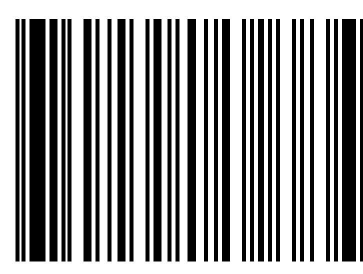
Der **Code 39 (Full ASCII)** ermöglicht die Verarbeitung des kompletten ASCII-Zeichensatzes mit dem Zeichensatz des Code 39.

Länge	Variabel
Zeichensatz	ASCII
Prüfziffer	Optional Modulo 43 Modulo 11 (Gewichtung 7) Modulo 10 (Luhn-Algorithmus)

Siehe auch

- Code 39

Code 93



Code 93 ist ein universeller, alphanumerischer Code und wurde aus dem Code 39 weiterentwickelt.

Durch die Nutzung von diversen Strich- und Lückenbreiten hat er eine höhere Informationsdichte.

Jedes Zeichen besteht aus neun Modulen, drei Strichen und drei Lücken.

Länge	Variabel
Zeichensatz	Ziffern 0-9 Großbuchstaben A-Z Sonderzeichen - . $ / + % Leerzeichen
Prüfziffer	Modulo 47

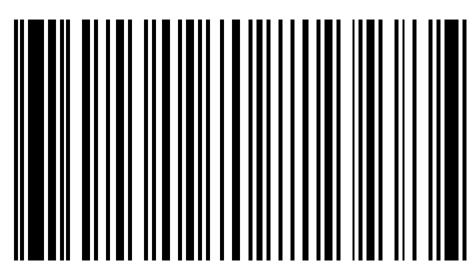
Siehe auch

- Code 93 (Full ASCII)

Code 93 (Full ASCII)


Hinweis

Dieser Barcode wird grafisch übertragen.



Der **Code 93 (Full ASCII)** ermöglicht die Verarbeitung des kompletten ASCII-Zeichensatzes mit dem Zeichensatz des Code 93.

Länge	Variabel
Zeichensatz	ASCII
Prüfziffer	Modulo 47

Siehe auch

- Code 93

Deutsche Post Identcode



Der **Identcode** wird von der Deutschen Post AG (DHL) verwendet und dient der automatischen Verteilung der Frachtsendung in den Postzentren. Der Identcode ist eine Anwendung des Code 2/5 Interleaved, jedoch mit einer anders berechneten Prüfziffer und einer abweichenden Formatierung der Klarschriftzeile.

Der **Identcode** dient zur eindeutigen Kennzeichnung eines Postpakets mit dessen Hilfe der Lauf eines Postpakets von der Annahme bis hin zur Auslieferung (Track & Trace) verfolgt werden kann.

Folgende Informationen sind im **Identcode** verschlüsselt:

- **Stelle 1-2:** Abgangsfrachtpostzentrum
- **Stelle 3-5:** Kundenkennung
- **Stelle 6-11:** Einlieferungsnummer
- **Stelle 12:** Prüfziffer

Länge	12
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- Deutsche Post Leitcode

Deutsche Post Leitcode



Der **Leitcode** wird von der Deutschen Post AG (DHL) verwendet und dient der automatischen Verteilung der Frachtsendung in den Postzentren. Der Leitcode ist eine Anwendung des Code 2/5 Interleaved, jedoch mit einer anders berechneten Prüfziffer und einer abweichenden Formatierung der Klarschriftzeile.

Folgende Informationen sind im **Leitcode** verschlüsselt:

- **Stelle 1-5:** Postleitzahl
- **Stelle 6-8:** Straßenkennzahl
- **Stelle 9-11:** Hausnummer
- **Stelle 12-13:** Produktcode
- **Stelle 14:** Prüfziffer

Länge	14
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- Deutsche Post Identcode

EAN-13, GTIN-13



Der **EAN-Code** wird hauptsächlich zur international eindeutigen Kennzeichnung von Produkten im Einzelhandel verwendet. Die Verpackungen der Erzeugnisse sind mit der GTIN - Globale Artikelidentnummer (engl. Global Trade Item Number, ehemals EAN - European Article Number) bedruckt.

Folgende Informationen sind im Code verschlüsselt, wobei die Stellen 7-9 die Basisnummer darstellen:

- **3 Stellen** - GS1-Länderpräfix (zum Beispiel 400 bis 440 für Deutschland, 760 bis 769 für die Schweiz und Liechtenstein, 900 bis 919 für Österreich)
- **4-6 Stellen** - Betriebsnummer
- **3-5 Stellen** - Artikelnummer (in Abhängigkeit von der Länge der Betriebsnummer)
- **1 Stelle** - Prüfziffer

Länge	13
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- EAN-13 + 2 Stellen
- EAN-13 + 5 Stellen
- EAN-8, GTIN-8
- Globale Artikelidentnummer (GTIN)

EAN-13 + 2 Stellen



EAN-13 mit zwei zusätzlichen Nutzzeichen.

Länge	15
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- EAN-13, GTIN-13
- EAN-13 + 5 Stellen

EAN-13 + 5 Stellen



EAN-13 mit fünf zusätzlichen Nutzzeichen.

Länge	18
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- EAN-13, GTIN-13
- EAN-13 + 2 Stellen

EAN-8, GTIN-8



Der **EAN-8** stellt die einfachste und kürzeste Form des **EAN-Codes** dar, da er auf 8 Stellen begrenzt ist. Er kommt dort zum Einsatz, wo nur wenig Platz für eine Warenauszeichnung zur Verfügung steht und ein EAN-13 mehr als 25% des Platzes auf der entsprechenden Produktseite einnehmen würde.

Folgende Informationen sind im Code verschlüsselt:

- **3 Stellen** - GS1-Länderpräfix (zum Beispiel 400 bis 440 für Deutschland, 760 bis 769 für die Schweiz und Liechtenstein, 900 bis 919 für Österreich)
- **4 Stellen** - Artikelnummer (vom Hersteller vergeben)
- **1 Stelle** - Prüfziffer

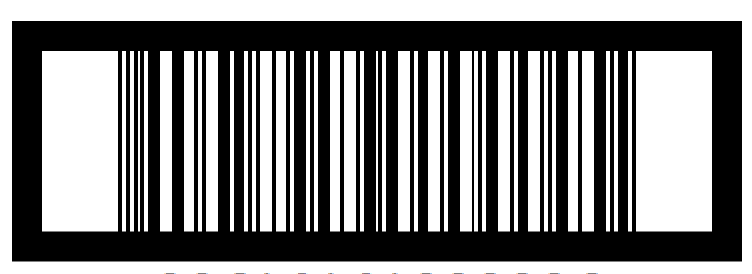
Eine GTIN-Kurznummer wird nur auf Antrag vergeben, da diese Nummern nur begrenzt verfügbar sind. Die mit einer 2 als Startziffer beginnenden **EAN-8-Codes** können innerhalb des eigenen Unternehmens frei verwendet werden, sind aber nicht weltweit eindeutig.

Länge	8
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- EAN-13, GTIN-13
- Globale Artikelidentnummer (GTIN)

ITF-14, SCC-14



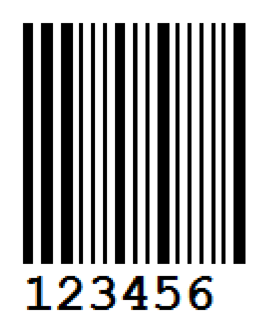
Mit dem **ITF-14** Barcode (auch SCC-14 genannt) wird in der Regel das Shipping Container Symbol dargestellt. Dieser Code wird für die Kennzeichnung von Kartons oder Paletten verwendet, welche Produkte enthalten, die mit einem EAN-13 gekennzeichnet sind.

Folgende Informationen sind im Code verschlüsselt:

- **1 Stelle** - Packungsart
- **12 Stellen** - Produktcode, i.d.R. die ersten 12 Ziffern des EAN-13
- **1 Stelle** - Prüfziffer

Länge	14
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Pharmacode

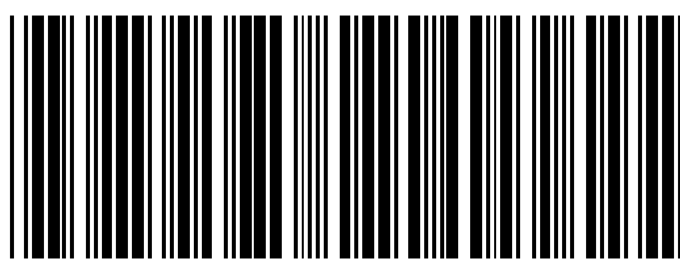


Der **Pharmacode** ist ein einfacher, numerischer Barcode der vom Unternehmen Laetus in Umlauf gebracht wurde. Er wird in der pharmazeutischen Industrie zur Packmittelkontrolle bzw. zur Steuerung von Verpackungsmaschinen verwendet.

Der **Pharmacode** der sowohl auf der Verpackung als auch auf dem Beipackzettel angebracht ist, sorgt dafür dass der richtige Beipackzettel in die dazugehörige Verpackung sortiert wird. Mit dem **Pharmacode** können nur Ganzzahlen von 3 bis 131070 kodiert werden.

Länge	Variabel
Zeichensatz	Ziffern 0-9
Prüfziffer	Keine

PZN



Die **Pharmazentralnummer (PZN)** ist ein in Deutschland bundeseinheitlicher Identifikationsschlüssel für Arzneimittel, Hilfsmittel und andere Apothekenprodukte. Sie ist eine achtstellige Nummer (7 Ziffern + Prüfziffer) mit vorangestelltem Minus-Zeichen, die Arzneimittel nach Bezeichnung, Darreichungsform, Wirkstoffstärke und Packungsgröße eindeutig kennzeichnet. Sie wird im Klartext (Zahlen) mit vorangestelltem „PZN" und als Strichcode (Code 39) auf jede Arzneimittelpackung aufgedruckt, wobei die Zeichenfolge „PZN" nicht im Strichcode enthalten ist.

Die **PZN** wird zentral von der Informationsstelle für Arzneispezialitäten (IFA) vergeben.

Im November 2010 gab die IFA die Erweiterung der **PZN** bekannt. Zum 1. Januar 2013 wurde die **PZN** auf 8 Stellen erweitert. Die bisherigen 7-stelligen **PZN** werden beibehalten und durch eine führende Null auf 8 Stellen erweitert. Neue **PZN** werden so lange mit führender Null vergeben, bis der alte Nummernkreis erschöpft ist. Die letzte Stelle bildet weiterhin die Prüfziffer.

Länge	7 bzw. 8
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 11

UPC-A, GTIN-12



Der **UPC-A** Code wird in den USA und Kanada zur Kennzeichnung von im Groß- und Einzelhandel angebotenen Gebrauchs- und Verbrauchsgüter verwendet.

Die EAN-13 ist kompatibel zum **UPC-A**, kodiert aber ein Zeichen mehr. Stellt man dem **UPC-A** eine führende Null voran, kann die Zahlenkette als gültige 13-stellige EAN benutzt werden.

Folgende Informationen sind im Code verschlüsselt:

- **1 Stelle** - Systemkennzeichen
- **5 Stellen** - Herstellernummer
- **5 Stellen** - Artikelnummer (vom Hersteller vergeben)
- **1 Stelle** - Prüfziffer

Länge	12
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- UPC-E
- Globale Artikelidentnummer (GTIN)

UPC-E



Der **UPC-E** Code ist eine komprimierte Version des UPC-A und wird überall dort eingesetzt, wo nur geringer Platz zur Verfügung steht. Durch die Methode der Nullunterdrückung kann eine 12-stellige UPC-A Nummer in eine 6-stellige UPC-E umgewandelt werden. Bei der Rückkonvertierung erfolgt eine Wiederauffüllung der Nullen und es entsteht somit wieder eine 12-stellige Komplettnummer.

Länge	8
Zeichensatz	Ziffern 0-9
Prüfziffer	Modulo 10

Siehe auch

- UPC-A

2D Barcodes

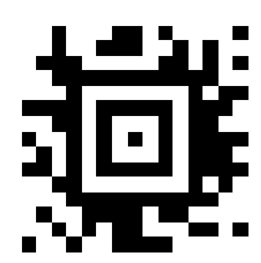
Zweidimensionale Barcodes kodieren die Informationen meistens in der Fläche, wobei die Informationen dann nicht in Form von Strichen, sondern in Form von (weißen und schwarzen) Punkten dargestellt werden. Es wird zwischen gestapelten Barcodes, Matrix-Codes, Punktcodes und einigen weiteren Sonderformen unterschieden.

Unterstützte Barcodes

- Aztec Code
- Codablock F
- DataMatrix
- MaxiCode
- PDF417
- QR-Code

Aztec Code

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **Aztec Code** ist ein 2D Matrix-Code in dessen Mittelpunkt sich das Suchelement befindet das aus mehreren auf einem Raster angeordneten quadratischen Modulen besteht. Der **Aztec Code** wird hauptsächlich im Transportwesen wie z.B. für Online Tickets der Deutschen Bahn verwendet.

Es können sehr kleine (ab 12 Zeichen) und große Datenmengen (bis 3067 alphanumerische Zeichen oder 3832 numerische Zeichen) kodiert werden.

Der **Aztec Code** besteht aus drei festen und zwei variablen Bestandteilen. Die festen Bestandteile sind: das zentrale Erkennungsmuster (Finder Pattern), das Ausrichtungsmuster (Orientation Patterns) und das Referenzraster (Reference Grid). Die Modusinformation (Mode Message) und die Datenschichten (Data Layers) sind die variablen Komponenten des Codes.

Der **Aztec Code** ist einer der wenigen Barcodes der keine Ruhezone benötigt. Dank der Reed-Solomon Fehlerkorrektur ist die Rekonstruktion des Dateninhalts selbst dann noch möglich wenn der Code bis zu 25% bei großen und bis zu 40% bei kleinen Codes zerstört worden sind.

Das sogenannte Core Symbol des **Aztec Codes** enthält das zentrale Erkennungsmuster, das Ausrichtungsmuster und die Modusinformation.

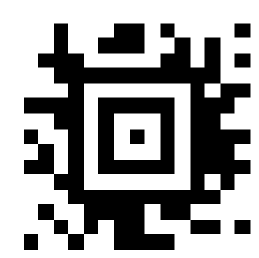
Länge	3067 alphanumerische Zeichen 3832 numerische Zeichen
Zeichensatz	ASCII
Prüfziffer	Intern

Siehe auch

- Aztec Runes

Aztec Runes

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **Aztec Code** ist ein 2D Matrix-Code in dessen Mittelpunkt sich das Suchelement befindet das aus mehreren auf einem Raster angeordneten quadratischen Modulen besteht. Der **Aztec Code** wird hauptsächlich im Transportwesen wie z.B. für Online Tickets der Deutschen Bahn verwendet.

Es können sehr kleine (ab 12 Zeichen) und große Datenmengen (bis 3067 alphanumerische Zeichen oder 3832 numerische Zeichen) kodiert werden.

Der **Aztec Code** besteht aus drei festen und zwei variablen Bestandteilen. Die festen Bestandteile sind: das zentrale Erkennungsmuster (Finder Pattern), das Ausrichtungsmuster (Orientation Patterns) und das Referenzraster (Reference Grid). Die Modusinformation (Mode Message) und die Datenschichten (Data Layers) sind die variablen Komponenten des Codes.

Der **Aztec Code** ist einer der wenigen Barcodes der keine Ruhezone benötigt. Dank der Reed-Solomon Fehlerkorrektur ist die Rekonstruktion des Dateninhalts selbst dann noch möglich wenn der Code bis zu 25% bei großen und bis zu 40% bei kleinen Codes zerstört worden sind. Das sogennante Core Symbol des **Aztec Codes** enthält das zentrale Erkennungsmuster, das Ausrichtungsmuster und die Modusinformation.

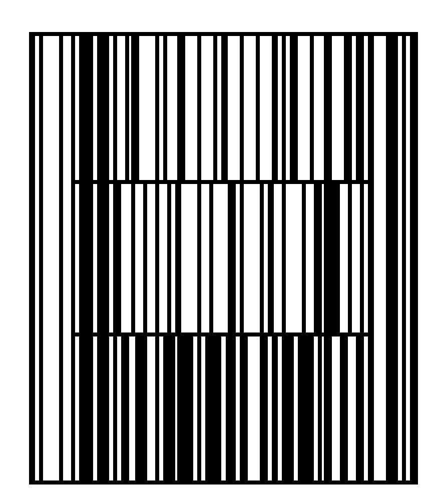
Länge	3067 alphanumerische Zeichen 3832 numerische Zeichen
Zeichensatz	ASCII
Prüfziffer	Intern

Siehe auch

- Aztec Code

Codablock F

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **Codablock F** ist ein 2D Barcode bei dem mehrere Code 128 übereinander gestapelt sind. Der Code wird hauptsächlich im Gesundheitswesen eingesetzt.

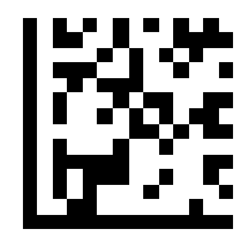
Mit dem **Codablock F** lassen sich 2 bis 44 Codezeilen darstellen. In jeder Zeile können vier bis 62 Zeichen codiert werden.

Das Prinzip des Codablock Barcodes arbeitet wie der Zeilenumbruch eines Texteditors. Ist eine Zeile voll, dann wird in die nächste Zeile umgebrochen, wobei zu jeder Zeile eine Zeilennummer kodiert und dem fertigen Block die Anzahl der Zeilen eingefügt wird. Somit enthält jede Zeile einen Indikator zur Orientierung für die Lesegeräte und der gesamte Code zwei Prüfzeichen um den Inhalt der Gesamtnachricht sicherzustellen.

Länge	In zwei bis 44 Zeilen können jeweils vier bis 62 Zeichen (maximal 2725 Zeichen) codiert werden.
Zeichensatz	ASCII
Prüfziffer	Intern

DataMatrix

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **DataMatrix** Code ist einer der bekanntesten 2D Barcodes. Mit dem **DataMatrix** Code können sehr viele Daten auf kleine Flächen kodiert werden. Aus diesem Grund wird er häufig für dauerhafte Direktbeschriftungen mittels Laser in der Produktion (Leiterplatten), mit Nadelprägung im Automobilbau, bei Analysegeräten und Instrumenten (Chemie, Medizin), aber auch zunehmend als gedrucktes Codebild im Dokumentenhandling (Tickets, DV-Freimachung beim Postversand) verwendet.

Der Code besteht aus quadratischen Modulen die innerhalb eines Erkennungsmuster (finder pattern) angeordnet sind. Die Module bestehen aus hellen und dunklen Zellen (schwarz = aktiv ->binär 1 / weiß = inaktiv -> binär 0) die aneinander anschließen.

Das Erkennungsmuster als Umfang des **DataMatrix** Codes hat die Breite eines Moduls und besteht an den linken und unteren Seiten aus zwei Linien, die nur dunkle Module enthalten. Die Linien an den rechten und oberen Seiten des Symbols werden abwechselnd aus dunklen und hellen Modulen dargestellt. Eine Ruhezone mit der Breite eines Moduls umgibt den Barcode.

Die einheitliche Symbolgröße und der feste Symbolabstand machen das Lesen und Dekodieren des Codes sehr sicher. Der **DataMatrix** wird aus den folgenden vier Komponenten gebildet:

- **Nutzdaten:** Suchelemente und Taktzellen umrahmen diesen Bereich der die redundande Daten für die Datensicherheit enthält.
- **Feste Begrenzungslinie (finder pattern):** Abgrenzung die für die Aufrichtung und Entzerrung des Codes verwendet wird, um jeden Lesewinkel zu ermöglichen.
- **Offene Grenzlinie (alternating pattern):** Gegenüberliegende Ecke der 'festen Begrenzungslinie'. Diese Linien befinden sich oben und auf der rechten Seite und bestehen aus weißen und schwarzen Punkten (offene Linien).
- **Ruhezone:** Bereich um den Code herum der keine Informationen oder Muster enthält. Dieser muss mindestens so breit sein, wie eine Spalte/Zeile bzw. der Punkt des Codes.

Zur Erstellung des **DataMatrix** wird die Reed-Solomon-Fehlerkorrektur ECC 200 verwendet. Mit dieser Fehlerkorrektur bleibt ein **DataMatrix** auch dann noch lesbar wenn bis zu 25% des Codes überdeckt oder zerstört sind.

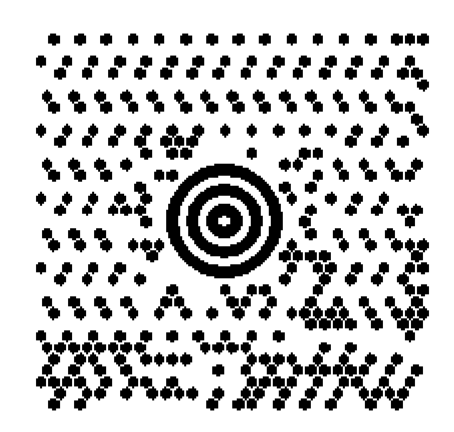
Länge	2335 alphanumerische Zeichen 3116 numerische Zeichen
Zeichensatz	ASCII
Prüfziffer	Intern

Siehe auch

- GS1 DataMatrix

MaxiCode

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **MaxiCode** ist ein 2D Barcode mit einer fixen Größe von 25,4 mm x 25,4 mm (1 in. x 1 in.). In diese Fläche von 645 mm^2 (1 sq in.) können die Nutzdaten codiert werden. Aufgebaut ist der Code aus 884 sechseckigen Modulen die ein Erkennungsmuster (Finder Pattern) aufweisen.

Der **MaxiCode** wurde von der Firma UPS zur schnellen Identifikation, Verfolgung und Sortierung von Paketen entwickelt und enthält die UPS Kontrollnummer, Gewicht, Art der Sendung und Adresse.

Der Code ist leicht erkennbar am bullaugenförmigen Suchmuster in der Mitte des Symbols. Durch die Reed-Solomon-Fehlerkorrektur ist eine Rekonstruktion des 2D Barcodes noch möglich selbst wenn bis zu 25% des Codes zerstört worden sind.

Labelstar Office unterstützt folgende Modi:

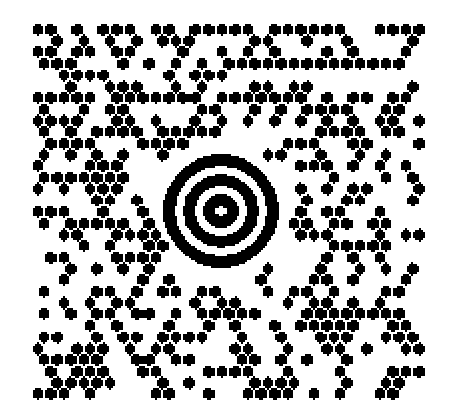
- **Mode 2:** Zustellernachricht (US Carrier)
- **Mode 3:** Zustellernachricht (International Carrier)
- **Mode 4:** Standardnachricht

Länge	93 alphanumerische Zeichen 138 numerische Zeichen
Zeichensatz	ASCII
Prüfziffer	Intern

Zustellernachricht

Feld	Beschreibung	Daten	Erforderlich	Beispiel
Primärdaten				
Postleitzahl (Empfänger)		Mode 2 (US Carrier): numerisch 9-stellig Mode 3 (International Carrier): alphanumerisch 6-stellig	ja	000012345
Ländercode	Ländercode des Empfängers nach ISO 3166.	numerisch 3-stellig	ja	276
Serviceklasse	Vom Spediteur vergebene Serviceklasse.	numerisch 3-stellig	ja	001
Sekundärdaten				
Auftragsnummer	Sendungsverfolgungnummer (UPS Tracking Number).	alphanumerisch 10- bzw. 11-stellig	ja	9A00001234
Beförderungscode	Standard Beförderungscode (Standard Carrier Alpha Code).	UPSN	ja	UPSN
UPS-Absendernummer	Beförderungsnummer (UPS Shipper Number).	alphanumerisch 6-stellig	ja	07X720
Tag der Abholung	Julianischer Tag im Jahr an dem das Paket abgeholt worden ist.	numerisch 3-stellig	ja	155
Versandnummer	Kundenspezifische Referenznummer.	alphanumerisch bis zu 30-stellig	-	
Paket n/x		numerisch bis 3 stellig/numerisch bis 3-stellig	ja	1/1
Paketgewicht	Paketgewicht, auf das nächste Pfund aufgerundet.	numerisch bis 3-stellig	ja	015
Adressvalidierung		Y bzw. N	ja	Y
Adresse (Empfänger)		alphanumerisch bis 35-stellig	-	Muster GmbH
Stadt (Empfänger)		alphanumerisch bis 20-stellig	ja	Musterstadt
Land (Empfänger)		alpha 2-stellig	ja	DE

Beispielcode



PDF417

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **PDF417** ist ein 2D Barcode der aus mehreren aufeinandergestapelten Zeilen auf einem rechteckigen Feld basiert. Verwendet wird der **PDF417** hauptsächlich auf Ausweisen, von Speditionen, in der Automobilindustrie und in Verwaltungsbehörden, z.B. von der Agentur für Arbeit um Manipulationen an Fragebögen zu verhindern.

Das Barcodesymbol besteht aus 3 bis 90 Zeilen und 1 bis 30 Spalten. Jede Zeile enthält eine linke und rechte Ruehezone (Quiet Zone), ein Start-/Stoppzeichen (Start/Stop Patterns), einen linken und rechten Zeilenindikator und 1 bis 30 Symbolzeichen (Symbol Characters). Ein **PDF417** Symbol besteht aus Barcodedaten, Prüf- und Korrekturzeichen. Die verwendeten Zeichen werden in Codewörtern kodiert. Ein Codewort besteht aus 17 Modulen die jeweils aus 4 Strichen und 4 Lücken gebildet werden.

Die Fehlerkorrektur wird mit dem Reed-Solomon Algorithmus in 9 wählbaren Sicherheitsstufen (Error Correction Levels) erfasst. Bei eingestellter Sicherheitsstufe 0 kann ein Fehler erkannt aber nicht korrigiert werden, mit Sicherheitsstufen 1 bis 8 können Fehler auch korrigiert werden.

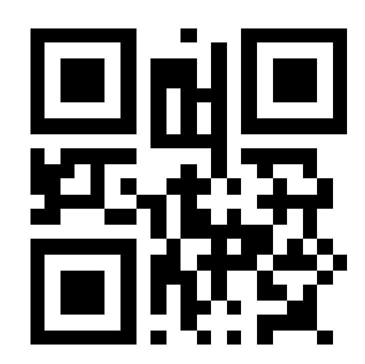
Verwendung der Fehlerkorrektur:

- **ECL 2:** weniger als 41 Codewörter
- **ECL 3:** 41 bis 160 Codewörter
- **ECL 4:** 161 bis 320 Codewörter
- **ECL 5:** mehr als 320 Codewörter

Länge	1850 alphanumerische Zeichen 2725 numerische Zeichen
Zeichensatz	ASCII
Prüfziffer	Intern

QR-Code

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **QR-Code** ist ein 2D Barcode aufgebaut aus quadratischen Modulen, angeordnet auf einem quadratischen Feld. Der **QR-Code** wird verbreitet in Bereich mobiler Endgeräte (z.B. Tablet PCs, Smartphones) und in der Industrie eingesetzt.

Der **QR-Code** erfordert eine Kamera und einen QR-Code Reader, um die kodierten Daten zu entschlüsseln. Die Kameras mit denen die meisten Mobiltelefone ausgestattet sind, erweitern den Einsatzbereich der **QR-Codes**, insbesondere im Hinblick auf die ständig steigende Nutzung von Smartphones.

Der Code besteht aus mehreren Funktionsmustern (Function Patterns) und einem kodierten Bereich (Encoding Region). Zu den Function Patterns gehören drei Erkennungsmuster (Finder Patterns), die Synchronisationsmuster (Timing Patterns), die Ausrichtungsmuster (Alignment Patterns) und die Abstandhalter (Separators).

Der kodierte Bereich enthält die eigentlichen Nutzdaten und Informationen über die Fehlerkorrektur, die Version und das Format des **QR-Codes**. Der gesamte **QR-Code** ist von einer Ruhezone (Quiet Zone) umgeben. Die Daten können in 40 Symbolgrößen kodiert werden. Die Symbolgrößen reichen von 21×21 bis zu 177×177 Modulen.

Mit **QR-Codes** können Informationen auf eine Vielzahl von Medien und Produkten gebracht und die Nutzung von Smartphones wesentlich erweitert werden. Räumliche Einschränkungen sind mit dem **QR-Code** vorbei. **QR-Codes** stellen Anbietern eine große Anzahl von Optionen zur Erweiterung ihrer Media/Marketing Möglichkeiten zur Verfügung.

Länge	4296 alphanumerische Zeichen 7089 numerische Zeichen
Zeichensatz	ASCII
Prüfziffer	Intern

Was kann ein QR-Code alles enthalten?

In einem QR-Code kann man Informationen unterschiedlichster Art speichern.

Mit **Labelstar Office** können Codes mit folgenden Inhalten erzeugt werden:

- **Beliebiger Text**
- **Webseite/Weblink** Scannt man den Code wird der Nutzer ohne lästiges Eintippen der Adresse auf eine Internetseite weitergeleitet.
- **Visitenkarte** Der Nutzer kann die Kontaktdaten mit einem „Klick“ seiner Kontakliste hinzufügen.
- **Veranstaltung/Event** Zum Beispiel Firmenjubiläum, Tag der offenen Tür oder Produktvorstellung. Ähnlich wie bei den Kontaktdaten, kann die Veranstaltung mit einem „Klick“ dem „Handy-Kalender“ hinzugefügt werden.
- **Telefonnummer** Mit einer im QR-Code gespeicherten Telefonnummer lässt sich am Handy ein Anruf direkt ausführen – sofern der Codeleser diese Aktion unterstützt.
- **E-Mail** Ein QR-Code kann eine komplette E-Mail inklusive Empfänger enthalten. Der Nutzer muss die E-Mail nicht schreiben, sondern versendet sie mit einem „Klick“. Mögliche Anwendungen: die Anforderung von Infos oder die Teilnahme an Gewinnspielen.
- **SMS** Eine SMS schreiben, um an einer Aktion teilzunehmen, ist zu umständlich? Es genügt auch, den QR-Code zu scannen. Dann erscheint die SMS im Display und lässt sich versenden.
- **Geo-Daten** Ein QR-Code verarbeitet auch Geo-Daten. Damit lässt sich der Ort etwa in Google Maps anzeigen.
- **WLAN-Zugangsdaten**

GS1 Barcodes

GS1 (Global Standards One) ist eine weltweite Organisation, die globale Standards zur Verbesserung von Wertschöpfungsketten gestaltet und umsetzt sowie weltweit für die Vergabe der GTIN - Globalen Artikelidentnummer (engl. Global Trade Item Number) zuständig ist. Zur Datenabgrenzung werden vordefinierte Application Identifiers verwendet.

Unterstützte Barcodes

- GS1 DataBar
- GS1 DataMatrix
- GS1-128

GS1 DataBar

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

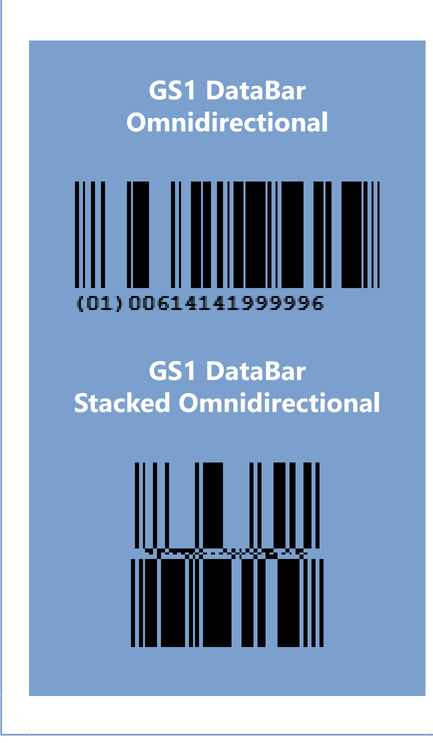
GS1 DataBar ist ein kleiner linearer Barcode, in dem Sie neben der GTIN-Artikelnummer Zusatzinformationen wie Gewicht oder Mindesthaltbarkeitsdatum auf kleinstem Raum verschlüsseln können. Der Strichcode ist lage- und richtungsunabhängig lesbar und daher für den Einsatz am Point of Sale (PoS) geeignet.

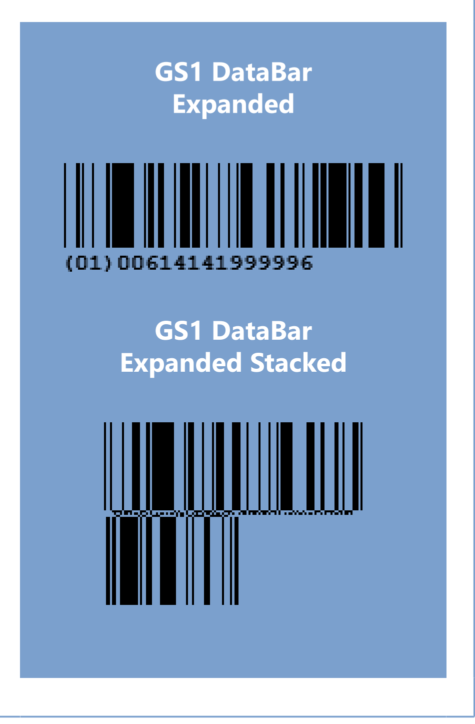
Der **GS1 DataBar** schließt Lücken, wo Artikel bislang nicht oder nur eingeschränkt gekennzeichnet werden konnten. Beispielsweise wird er genutzt, um gewichtsvariable Ware wie Obst oder Käse zu kennzeichnen. Er ist jedoch auch für Frischeprodukte allgemein oder Gutscheine geeignet.

Seit 2010 wird der **GS1 DataBar** auf Basis bilateraler Absprachen zwischen Handel und Lieferanten am PoS angewendet. Bis spätestens 2014 sollen jedoch alle Kassensysteme in der Lage sein, den **GS1 DataBar** zu scannen.

Die Leistungsfähigkeit des **GS1 DataBar** ist enorm vielseitig, da es sich um eine Strichcodefamilie handelt. **GS1 DataBar Expanded** oder **GS1 DataBar Expanded Stacked** verschlüsseln beispielsweise bis zu 74 numerische und 41 alphanumerische Zeichen. Vier der insgesamt sieben Ausprägungen sind richtungs- und lageunabhängig lesbar und daher ab 2010 für den PoS zugelassen.

Omnidirektionale GS1 DataBar-Symbole "PoS-kompatibel"



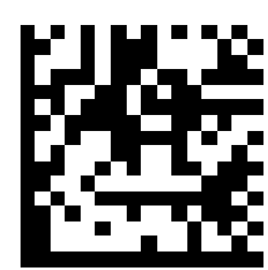


Sehr kleine GS1-DataBar-Symbole nicht "PoS-kompatibel"



GS1 DataMatrix

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **GS1 DataMatrix** ist eine Sonderform des DataMatrix und wird vor allem im Handel und der Industrie, hauptsächlich bei der Waren- und Palettenauszeichnung eingesetzt.

Er ist ein zweidimensionaler Code, in dem viele Informationen fälschungssicher auf sehr kleinem Platz verschlüsselt werden können. Eine GTIN kann beispielsweise schon auf einer Fläche von 5 x 5 mm dargestellt werden. Somit eignet sich dieser Barcode, um Kleinstprodukte und sogar einzelne Bauteile von Produkten zu kennzeichnen.

Der **GS1 DataMatrix** verwendet Application Identifiers zur Datenabgrenzung, um unterschiedlichste Informationen geschützt zu verschlüsseln.

Länge	Variabel
Zeichensatz	ASCII
Prüfziffer	Keine

Siehe auch

- DataMatrix

GS1-128

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	



Der **GS1-128** ist eine Sonderform des Code 128 und wird vor allem im Handel zur Waren- und Palettenauszeichung verwendet. Bis 2009 hieß der Standard UCC/EAN-128 und wurde unter anderem auch einfach EAN 128 genannt.

Die Länge des **GS1-128** ist variabel, sollte jedoch die maximale Länge von 165 mm nicht übersteigen. Es dürfen insgesamt maximal 48 Nutzzeichen inklusive der Application Identifiers und FNC1 Trennzeichen kodiert werden.

Länge	Variabel
Zeichensatz	ASCII
Prüfziffer	Modulo 103

Siehe auch

- Code 128

Prüfziffernberechnung

Eine Prüfziffer ist die einfachste Form einer Prüfsumme. Die Prüfziffer wird nach einer bestimmten Rechenvorschrift aus den übrigen Zeichen und Ziffern berechnet. Durch Berechnung und Vergleich der Prüfziffer können Eingabefehler erkannt werden. In einem Datenerfassungsgerät kann bei einer abweichenden Prüfziffer die Eingabe verworfen werden.

Siehe auch

- Modulo 10
- Modulo 10 (Luhn Algorithmus)
- Modulo 11

Modulo 10

Eine Prüfziffer nach Modulo 10 wird u.a. von EAN Barcodes verwendet, z.B. EAN-13, GTIN-13.

Die Berechnung der Prüfziffer erfolgt nach Modulo 10 mit der Gewichtung 3. Bei der Berechnung wird mit der ersten Nutzziffer von rechts mit dem Gewichtungsfaktor 3 begonnen. Die Einzelprodukte werden zu einer Summe addiert. Die Differenz zwischen dem Produkt und dem nächsten vollen "Zehner" (Aufrundung) ergibt die Prüfziffer.

Beispiel



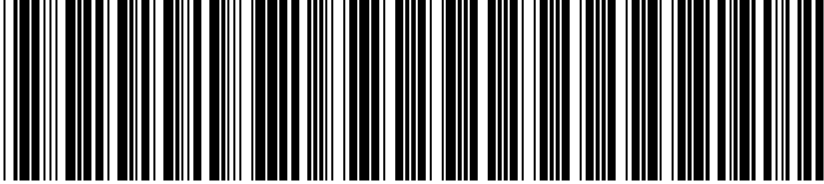
Ziffernstellen:	13	12	11	10	9	8	7	6	5	4	3	2	1
Klartext:	4	0	1	2	3	4	5	9	8	7	6	5	2
Prüfziffer:	2												
Nutzziffernfolge:	4	0	1	2	3	4	5	9	8	7	6	5	
Gewichtungsfaktoren:	1	3	1	3	1	3	1	3	1	3	1	3	
Einzelprodukte:	4	0	1	6	3	12	5	27	8	21	6	15	
Summe der Einzelprodukte:	4 + 0 + 1 + 6 + 3 + 12 + 5 + 27 + 8 + 21 + 6 + 15 = 108												
Modulo 10:	108 Mod. 10 = 8 (108/10 = 10 Rest 8)												
Differenz zu 10 ergibt die Prüfziffer:	10 - 8 = 2												
Prüfziffer:	2												

Modulo 10 (Luhn-Algorithmus)

Der Luhn-Algorithmus oder die Luhn-Formel wurde in den 1960er Jahren von Hans Peter Luhn entwickelt.

Die Berechnung der Prüfziffer erfolgt nach Modulo 10 mit der Gewichtung 2. Bei der Berechnung wird mit der ersten Nutzziffer von rechts mit dem Gewichtungsfaktor 2 begonnen. Sofern das Produkt größer als 9 ist wird die Quersumme gebildet, was denselben Effekt hat als würde man 9 subtrahieren. Die Einzelprodukte werden zu einer Summe addiert. Das Ergebnis wird dann durch 10 dividiert. Der daraus resultierende Rest (Modulo 10) ist die Prüfziffer.

Beispiel



Ziffernstellen:	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Klartext:	4	5	5	6	7	3	7	5	8	6	8	9	9	8	5	5
Prüfziffer:	5															
Nutzziffernfolge:	4	5	5	6	7	3	7	5	8	6	8	9	9	8	5	
Gewichtungsfaktoren:	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	
Einzelprodukte:	8	5	10	6	14	3	14	5	16	6	16	9	18	8	10	
Quersumme:	8	5	1	6	5	3	5	5	7	6	7	9	9	8	1	
Summe der Quersummen:	8 + 5 + 1 + 6 + 5 + 3 + 5 + 5 + 7 + 6 + 7 + 6 + 9 + 9 + 8 + 1 = 85															
Modulo 10:	85 Mod. 10 = 8 (85/10 = 8 Rest 5)															
Prüfziffer:	5															

Modulo 11

Eine Prüfziffer nach Modulo 11 wird z.B. vom PZN verwendet.

Bei der Berechnung eine Prüfziffer nach Modulo 11 werde alle Stellen mit einem Gewichtungsfaktor multipliziert und die Ergebnisse addiert. Der Gewichtungsfaktor entspricht der Position der Ziffer + 1 (d.h. 1. Stelle * 2, 2. Stelle * 3 ... 6. Stelle * 7). Das Ergebnis wird dann durch 11 dividiert. Der daraus resultierende Rest (Modulo 11) ist die Prüfziffer.

Da die 11 eine Primzahl ist, verspricht man sich von diesen Verfahren sehr gute Fehlererkennungseigenschaften.

Beispiel



Ziffernstellen:	1	2	3	4	5	6	7
Klartext:	6	3	1	9	4	2	9
Prüfziffer:	9						
Nutzziffernfolge:	6	3	1	9	4	2	
Gewichtungsfaktoren:	2	3	4	5	6	7	
Einzelprodukte:	12	9	4	45	24	14	
Summe der Einzelprodukte:	12 + 9 + 4 + 45 + 24 + 14 = 108						
Modulo 11:	108 Mod. 11 = 9 (108/11 = 9 Rest 9)						
Prüfziffer:	9						

Globale Artikelidentnummer (GTIN)

Die **GTIN - Globale Artikelidentnummer** (engl. Global Trade Item Number, ehemals EAN - European Article Number) ist eine Identifikationsnummer durch die jeder Artikel oder jede Dienstleistung weltweit überschneidungsfrei identifiziert werden kann. Sie wurde für die Verwendung im Bereich der elektronischen Datenverarbeitung konzipiert. Versorgungs- und Lieferprozesse lassen sich mit der **GTIN** zielgerichtet steuern. Sie ist zentraler Baustein der Warenwirtschaft.

Die **GTIN** fungiert als Zugriffsschlüssel auf die in Datenbanken hinterlegten Produktinformationen, wie Bezeichnung, Gewicht, Gebindegröße oder Warengruppe. Normalerweise umfasst die **GTIN** 13 Stellen. Für kleine Artikel, auf denen die lange Nummer nicht untergebracht werden kann, steht eine 8-stellige Kurznummer zur Verfügung.

Die **GTIN** kann 8, 12, 13 oder 14 Stellen lang sein. Jede **GTIN** ist eindeutig und kann einem bestimmten Unternehmen und Produkt zugeordnet werden.

Bezeichnung	Frühere Bezeichnung
GTIN-8	EAN-8
GTIN-12	UPC-A
GTIN-13	EAN-13
GTIN-14	-

Die 14-stellige **GTIN** wird durch Voranstellen führender Nullen aus den bisherigen 8-, 12- und 13-stelligen Artikelnummern gebildet:

GTIN-Typen	GTIN-Format													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GTIN-8	0	0	0	0	0	0	N	N	N	N	N	N	N	C
GTIN-12	0	0	N	N	N	N	N	N	N	N	N	N	N	C
GTIN-13	0	N	N	N	N	N	N	N	N	N	N	N	N	C
GTIN-14	N	N	N	N	N	N	N	N	N	N	N	N	N	C

Datenbanken

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Eine Fülle von Daten, die sich außerhalb eines Etiketts befinden, können innerhalb des Etiketts verwendet werden. Aber wie finden und importieren Sie diese Daten in **Labelstar Office**? Die Antwort ist ganz einfach: Sie müssen nur eine Datenverbindung erstellen und nutzen.

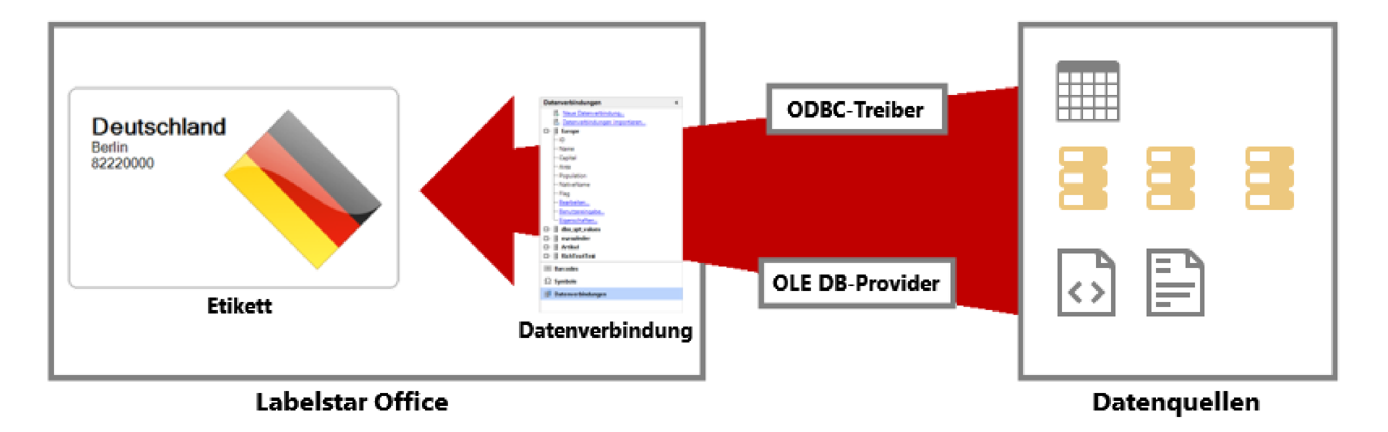
Die Daten auf einem Etikett können aus zwei verschiedenen Speicherorten stammen. Die Daten können direkt in dem Etikett gespeichert sein oder aus einer externen Datenquelle stammen, z. B. aus einer Textdatei oder einer Datenbank. Diese externe Datenquelle wird über eine Datenverbindung mit dem Etikett verbunden. Diese Datenverbindung ist eine Informationssammlung, die beschreibt, wie die externe Datenquelle gesucht wird bzw. wie der Anmelde- oder Zugriffsvorgang erfolgt.

Der Hauptvorteil der Verbindung mit externen Daten besteht darin, dass Sie diese Daten ändern können, ohne die Daten wiederholt in Ihr Etikett kopieren zu müssen - ein Vorgang, der zeitaufwendig und fehlerträchtig sein kann.

Um externe Daten in **Labelstar Office** zu verwenden, müssen Sie Zugriff auf die Daten besitzen. Wenn sich die externe Datenquelle, auf die Sie zugreifen möchten, nicht auf dem lokalen Computer befindet, müssen Sie sich gegebenenfalls an den Datenbankadministrator wenden, um das Kennwort, die Benutzerberechtigungen oder andere Verbindungsinformationen zu erhalten. Wenn die Datenquelle eine Datenbank ist, stellen Sie sicher, dass die Datenbank nicht im exklusiven Modus geöffnet ist. Wenn es sich bei der Datenquelle um eine Textdatei oder eine Kalkulationstabelle handelt, stellen Sie sicher, dass kein anderer Benutzer diese Tabelle für den exklusiven Zugriff geöffnet hat.

Viele Datenquellen erfordern außerdem einen ODBC-Treiber oder einen OLE DB-Provider, um den Datenfluss zwischen **Labelstar Office**, der Verbindungsdatei und der Datenquelle zu koordinieren.

Die folgende Abbildung fasst die wichtigsten Punkte zu Datenverbindungen zusammen.



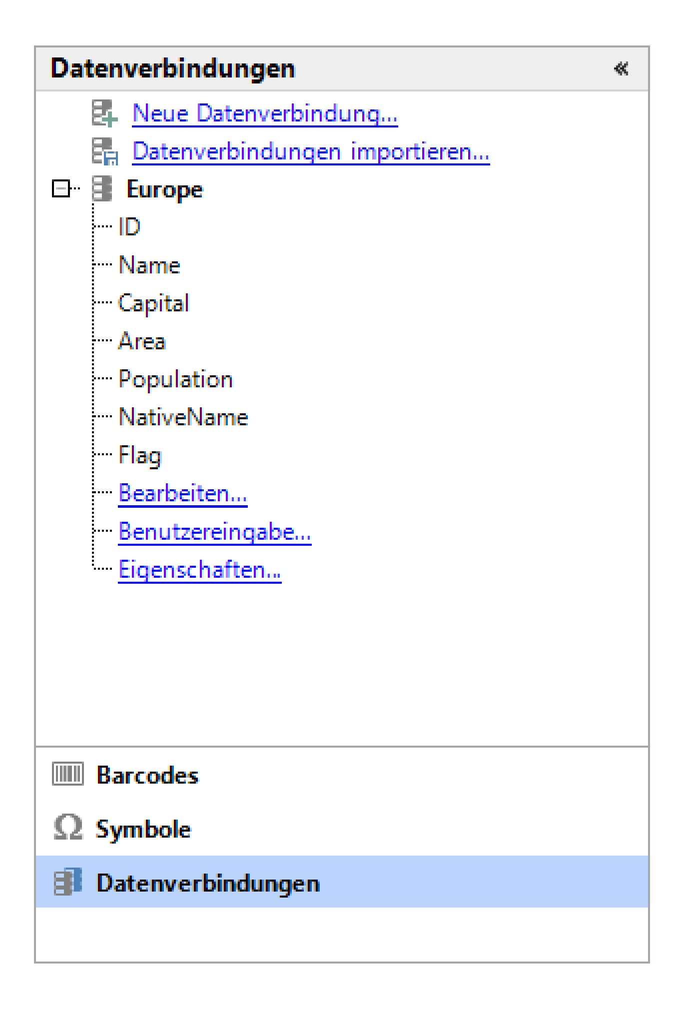
Erste Schritte

- Neue Datenverbindung anlegen
- Datenbanketikett erstellen

Neue Datenverbindung anlegen

Um eine neue Datenverbindung anzulegen, gehen Sie bitte folgendermaßen vor:

1. Aktivieren Sie die Ansicht **Datenverbindungen**.
2. Klicken Sie auf **Neue Datenverbindung**.
   Der **Datenverbindungs-Assistent** wird geöffnet.
3. Wählen Sie die Datenquelle aus, die Sie verwenden möchten.
4. Folgen Sie den Anweisungen im Assistenten.
5. Nach erfolgreicher Definition wird die neue Datenverbindung in der Liste angezeigt und die zugehörigen Datenbankfelder können auf einem Etikett verwendet werden.



Datenbanketikett erstellen

Ein Beispiel, wie Sie ein Datenbanketikett erstellen können, finden Sie in unseren Videoanleitungen.

Die im Video verwendeten Beispieldaten finden Sie im Verzeichnis: *%InstallDir%\Samples\Database*.

Europe.accdb Datenbank (Microsoft Access-Format)
Europe.lbex Etikettendefinition
Europe.txt Datenbank (Text-Format)
Europe.xml Datenbank (XML-Format)
Import Europe_accdv data connection.lbdx Importdatei für die Datenverbindung *Europe* (Microsoft Access-Format)
Import Europe_txt data connection.lbdx Importdatei für die Datenverbindung *Europe* (Text-Format)
Import Europe_xml data connection.lbdx Importdatei für die Datenverbindung *Europe* (XML-Format)



Protokollierung

Benötigte Programmvariante	PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Mit Hilfe der Protokollierung können Sie nachvollziehen, welche Daten wann, von wem und auf welchem Drucker gedruckt worden sind.

Welche Informationen werden bei der Protokollierung gespeichert?

Die in **Labelstar Office** enthaltene Protokollierungsoption protokolliert die folgenden Features:

- Datum/Zeitpunkt des Druckvorgangs
- Druckanzahl
- Seitenname
- Etikettenname
- Druckername
- Benutzername
- Feldinhalte

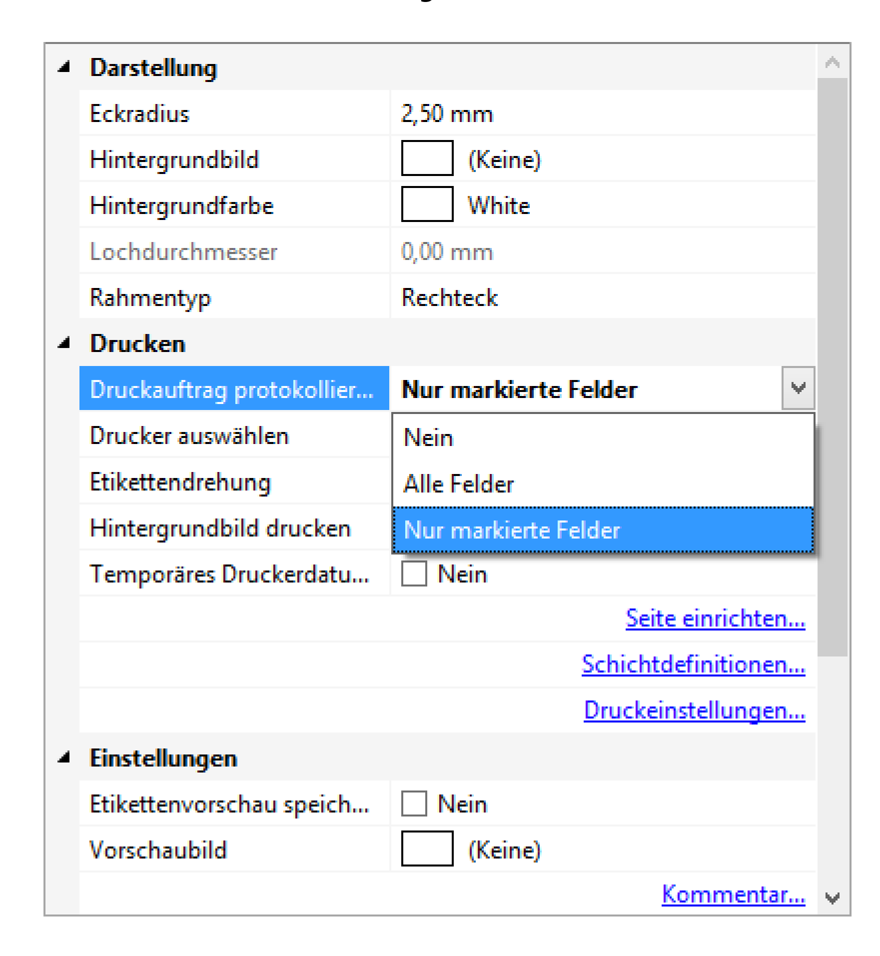
Siehe auch

- Aktivieren und Deaktivieren der Protokollierung
- Speicherort der Protokolldateien
- Registerkarte «Protokollierung»

Aktivieren und Deaktivieren der Protokollierung

So aktivieren und deaktivieren Sie die Protokollierung

1. Aktivieren Sie die **Etiketteneigenschaften** und klicken Sie auf **Druckauftrag protokollieren**.



2. Zum Aktivieren der Protokollierung wählen Sie eine der folgenden Optionen aus:
   - **Alle Felder** Es werden alle Feldinhalte protokolliert.
   - **Nur markierte Felder** Es werden nur die Inhalte der Felder protokolliert, bei denen die Option **Protokollieren** aktiviert ist.
3. Zum Deaktivieren der Protokollierung wählen Sie **Nein** aus.

Siehe auch

› Registerkarte «Protokollierung»

Speicherort der Protokolldateien

In der Registerkarte «Protokollierung» können Sie den Speicherpfad der Protokolldateien festlegen. Wählen Sie nur einen Pfad aus, auf den alle Benutzer Zugriff haben.



Hinweis

Wählen Sie niemals nur *C:* oder *C:\Windows* aus. Wenn überhaupt, dann erstellen Sie bitte einen neuen Ordner und lassen das Programm dort seine Protokolldateien ablegen (z.B. *C:\Log*).

Markup-Tags

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Mit Hilfe von Markup-Tags können Formatierungsanweisungen in den Text eingefügt werden.

Hinweis

Markup-Tags müssen richtig formatiert sein. Das bedeutet, dass alle Tags richtig geschlossen und alle Attributwerte in Hochkommas eingeschlossen werden müssen.

Die folgende Tabelle beschreibt die Formatierungsanweisungen.

Markup-Tag	Beschreibung	Beispiele
b	Der Text wird fett ausgegeben.	Fetter Text -> **Fetter Text**
br	Zeilenumbruch	Zeile 1 Zeile 2 -> Zeile 1 Zeile 2
em	Markiert hervorgehobenen Text. Der Text wird kursiv ausgegeben.	Hervorgehobener Text -> *Hervorgehobener Text*
font	Definiert die Schriftgröße, den Schriftnamen und die Schriftfarbe. *Arribute:* • ***size**: Schriftgröße* • ***name**: Schriftname* • ***color**: Schriftfarbe*	Beispieltext -> Beispieltext Beispieltext -> Beispiel**text**
i	Der Text wird kursiv ausgeben.	<i>Kusiver Text</i> -> *Kusiver Text*
rtl	Der Text wird von rechts nach links ausgegeben.	<rtl>Beispieltext</rtl> -> txetleipsieB
shadow	Der Text wird mit einem Schatten ausgegeben. *Attribute:* • ***color**: Schattenfarbe (Standard: Schwarz)* • ***offset**: Schattenoffset (Standard: 1,1)* • ***strength**: Schattenstärke (Standard: 1,1)* • ***style**: Schattenstil (Standard: Solid)* Solid Blurred (Verschwommen)	<shadow style='Blurred'>Beispieltext</shadow> -> Beispieltext <shadow color='Black'>Beispieltext</shadow> -> Beispieltext
strike	Der Text wird durchgestrichen ausgegeben.	<strike>Durchgestrichener Text</strike> -> ~~Durchgestrichener Text~~
stroke	Der Text wird mit einem Rahmen ausgegeben. *Attribute:* • ***width**: Rahmenstärke (Standard: 1)* • ***color**: Rahmenfarbe (Standard: Schwarz)*	<stroke color='#FF0000'>Beispieltext</stroke> -> Beispieltext

strong	Markiert besonders wichtigen (stark hervorgehobenen) Text. Der Text wird fett ausgegeben.	Stark hervorgehobener Text -> **Stark hervorgehobener Text**
sub	Tiefergestellter Text	H₂O -> H_2O
sup	Hochgestellter Text	Fußnote¹ -> Fußnote^1
u	Der Text wird unterstrichen ausgegeben.	<u>Unterstrichener Text</u> -> <u>Unterstrichener Text</u>

Zusätzlich unterstützte Sonderzeichen:

Zeichen	Beschreibung	Code
"	Anführungszeichen	"
'	Hochkomma	'
&	Kaufmännisches Und	&
<	Öffnende spitze Klammer	<
>	Schließende spitze Klammer	>
	Geschütztes Leerzeichen	
©	Copyright-Zeichen	©
®	Registrierte Marke	®
™	Warenzeichen	™

Allergenkennzeichnung von Lebensmitteln

Ab dem 13. Dezember 2014 müssen die Vorschriften der EU-Verordnung 1169/2011 betreffend die Information der Verbraucher über Lebensmittel (Lebensmittel-Informationsverordnung oder LMIV) eingehalten werden.

Die LMIV sorgt für Vorgaben zur besseren Lesbarkeit (unter anderem eine Mindestschriftgröße), eine klarere Kennzeichnung von Lebensmittelimitaten, eine verbesserte Allergenkennzeichnung vorverpackter Lebensmittel und die obligatorische Allergeninformation bei loser Ware sowie ab Dezember 2016 eine verpflichtende Nährwertkennzeichnung.

Folgende Allergene sowie daraus hergestellte Erzeugnisse müssen kenntlich gemacht werden:

- **Glutenhaltiges Getreide** (z.B. Weizen, Roggen, Gerste, Hafer, Dinkel, Kamut, Emmer, Einkorn, Grünkern oder Hybridstämme davon)
- **Krebstiere** (z.B. Krebs, Shrimps, Garnelen)
- **Eier** (z.B. als Flüssigei, Lecithin, (Ov)-Albumin)
- **Fische** (alle Fischarten)
- **Erdnüsse** (z.B. Erdnussöl, -butter)
- **Sojabohnen** (z.B. als Miso, Sojasoße, Sojaöl)
- **Milch** (Butter, Käse, Laktose, Molkenprotein)
- **Schalenfrüchte** (Mandeln, Haselnüsse, Walnüsse, Kaschunüsse, Pecannüsse, Pistazien, Macadamianüsse)
- **Sellerie** (Bleich-, Knollen- und Staudensellerie)
- **Senf** (z.B. Sendkörner und Senfpulver)
- **Sesamsamen** (z.B. als Sesamöl, Tahin, Gomasio)
- **Schwefeldioxid und Sulfite** (E 220- E 228, >10mg/kg or 10mg/l)
- **Lupinen** (z.B. als Lupinenmehl in glutenfreien Produkten und Lupineneinweiß in vegetarischen Produkten)
- **Weichtiere** (z.B. Schnecken, Tintenfisch, Austern, Muscheln)

Beispiel

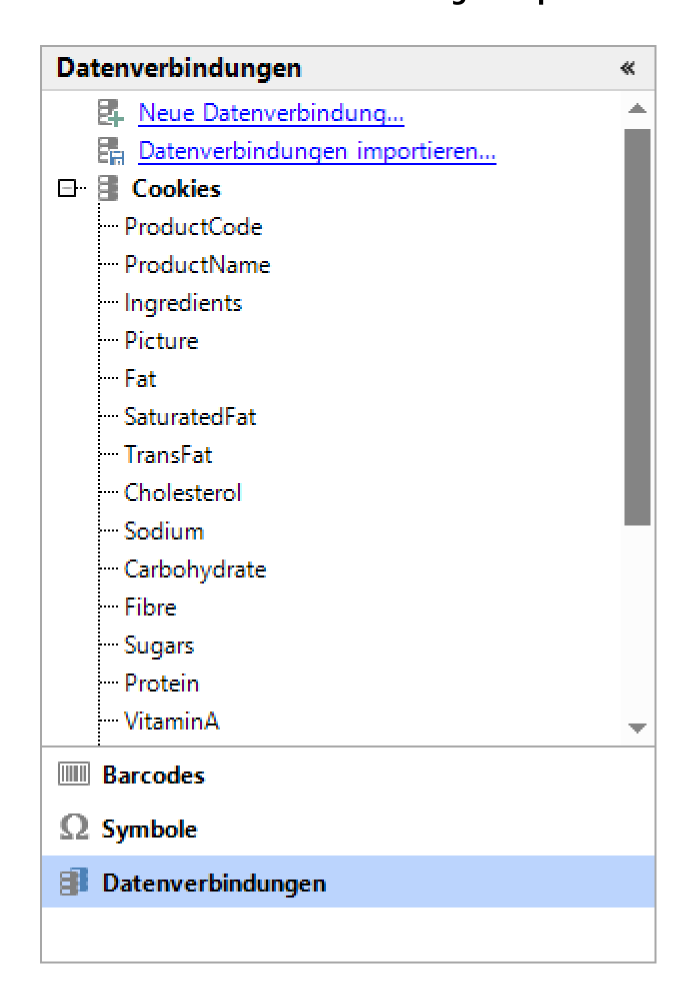
In diesem Beispiel wird gezeigt, wie Sie ein Datenbanketikett, unter Verwendung der Variable $ReplacePattern, definieren können, um bestimmte Wörter automatisch hervorzuheben.

Das Beispiel finden Sie im Verzeichnis: *%InstallDir%\Samples\Allergens*.

Allergens.txt Diese Datei enthält die Liste der Allergene, die hervorgehoben werden sollen.
Cookies.accdb Microsoft Access-Datenbank mit den Zutatenlisten.
Cookies.lbex Etikettendefinition
Import Cookies data connection.lbdx Importdatei für die Datenverbindung *Cookies*.

Um das Beispieletikett zu öffnen, gehen Sie bitte folgendermaßen vor:

1. Öffnen Sie **Labelstar Office**.
2. Aktivieren Sie die Ansicht **Datenverbindungen**.
3. Klicken Sie auf **Datenverbindungen importieren** und wählen Sie die Datei **Import Cookies data connection.lbdx** aus.



4. Öffnen Sie das Etikett **Cookies.lbex**.

Nutrition Facts

Per 1 cookie (28g)

Amount	% Daily Value
Calories: 139	
Fat 7 g	11%
Satured Fat 3 g	15%
+ Trans Fat 0 g	
Colesterol 45 mg	15%
Sodium 75 mg	3%
Carbohydrate 17 g	6%
Fibre 0 g	0%
Sugars 9 g	
Protein 2 g	

Vitamin A 2% Vitamin C 0%
Calcium 2% Iron 4%



Chocolate Cookie

INGREDIENTS: organic pastry flour (organic whole grain white **wheat**), organic evaporated cane juice, organic butter (cream, salt), organic dark chocolate chips (organic cacao mass, organic evaporated cane juice, organic cacao butter, may contain non-GMO **soy lecithin**), organic whole **eggs**, organic sunflower oil, organic vanilla extract, organic molasses, baking powder, baking soda, sea salt.

Unterstützte Grafik- und Vektorformate

Hinweis
Welche Formate unterstützt werden hängt von der verwendeten Programmvariante ab. Weitere Informationen finden Sie unter Programmvarianten.

- **ANIMATED GIF** - Graphics Interchange Format
- **BMP** - Standard Windows Bitmap
- **CUT** - Dr. Halo/Dr. Genius Clipboard Format
- **DDS** - Microsoft DirectDraw Surface Format
- **DIB** - Standard Windows Bitmap Format
- **PCD** - Kodak Photo-CD file
- **PCT, PICT, PIC** - Macintosh PICT Format
- **PCX** - PC Paintbrush Format
- **PDF/A** - Document Format for long term preservation
- **DICOM** - Digital Imaging and Communications in Medicine
- **EMF** - Enhanced Windows Metaformat
- **EXIF** - Exchangable Image Format
- **EXR** - OpenEXR Format
- **FAX, G3** - Group 3 Raw Fax Format
- **GIF, Interlaced GIF** - Graphics Interchange Format
- **HDR** - High Dynamic Range Format
- **IFF** - Interchange Format
- **ICO (single and multi page)** - Icone Format
- **J2K, J2C** - JPEG-2000 Codestream
- **JB2, JBIG2** - Joint Bi-level Image Experts Group
- **JIF, JFIF** - JPEG File Interchange Format
- **JNG** - JPEG Network Graphics
- **JP2** - JPEG-2000 Format
- **JPEG, JPG, JPE** - Joint Pointgraphic Expert Group
- **JPEG progressive**
- **KOA** - KOALA Format
- **LBM** - Interchange File Format-Interleaved Bitmap
- **MNG** - Multiple-image Network Graphics
- **PBM** - Portable Bitmap File
- **PBM Raw** - Portable Bitmap BINARY
- **PDF Multi-page** - Portable Document Format
- **PFM** - Portable Float Map
- **PGM** - Portable Graymap BINARY
- **PGM RAW** - Protable Graymap File
- **PSD** - Photoshop File
- **PNG** - Portable Network Graphics Format
- **PNM** - Portable Any Map
- **PPM** - Portable Pixmap File
- **PPM RAW** - Portable Pixmap BINARY

- **RAS** - Sun Raster Format
- **RAW camera image**
- **RAW memory bits** - RAW bitmap
- **RLE** - Standard Windows Bitmap format
- **SGI** - Silicon Graphics Image Format
- **TGA, TARGA** - TARGA Image Format
- **TIFF, TIF** - Tagged Image Format
- **TIFF Multi-page** - Multi-page Tagged Image Format
- **WBMP, WAP, WBM** - Wireless Bitmap
- **WEBP** - WebP Image Format
- **WMF** - Standard Windows Metaformat
- **XBM** - X Bitmap Format
- **XPM** - X Pixmap Format

Programmoptionen

In diesem Dialogfeld können Sie verschiedene Grundeinstellungen vornehmen und das Programm an Ihre persönlichen Vorlieben anpassen.

Um die Programmoptionen zu ändern, gehen Sie bitte folgendermaßen vor:

1. Klicken Sie in der Registerkarte **Datei** auf die Schaltfläche **Optionen**.
   Das Dialogfeld **Optionen** wird geöffnet.
2. Ändern Sie die gewünschten Einstellungen.
3. Klicken Sie auf die Schaltfläche **OK**, um die geänderten Einstellungen zu speichern.

Siehe auch

- Registerkarte «Allgemein»
- Registerkarte «Drucken»
- Registerkarte «Etikettenvorschau»
- Registerkarte «Memory Card»
- Registerkarte «Protokollierung»
- Registerkarte «Dateiablage»

Registerkarte «Allgemein»

In dieser Registerkarten können Sie verschiedene allgemeine Grundeinstellungen ändern.

Unter anderem können Sie auswählen, wie sich **Labelstar Office** beim Programmstart verhält:

- **Leeres Etikett** Es wird immer ein leeres Etikett geöffnet.
- **Zuletzt geöffnetes Etikett** Es wird das zuletzt geöffnete Etikett angezeigt.
- **Etikett laden** Es wird in bestimmtes Etikett angezeigt. Klicken Sie auf , um eine Datei auszuwählen.
- **Dialogfeld "Datei öffnen" anzeigen** Das Dialogfeld "Datei öffnen" wird angezeigt und es kann eine Datei ausgewählt werden.

Registerkarte «Drucken»

In dieser Registerkarte können Sie verschiedene Druckoptionen ändern.

Labelstar Office verwendet anfangs den Windows-Standarddrucker, aber Sie können einen anderen Standarddrucker für die Druckausgabe auswählen. Der Windows-Standarddrucker und der **Labelstar Office**-Standarddrucker sind unabhängig voneinander. Wenn Sie einen der beiden Standards ändern, beeinflusst dies den anderen nicht.

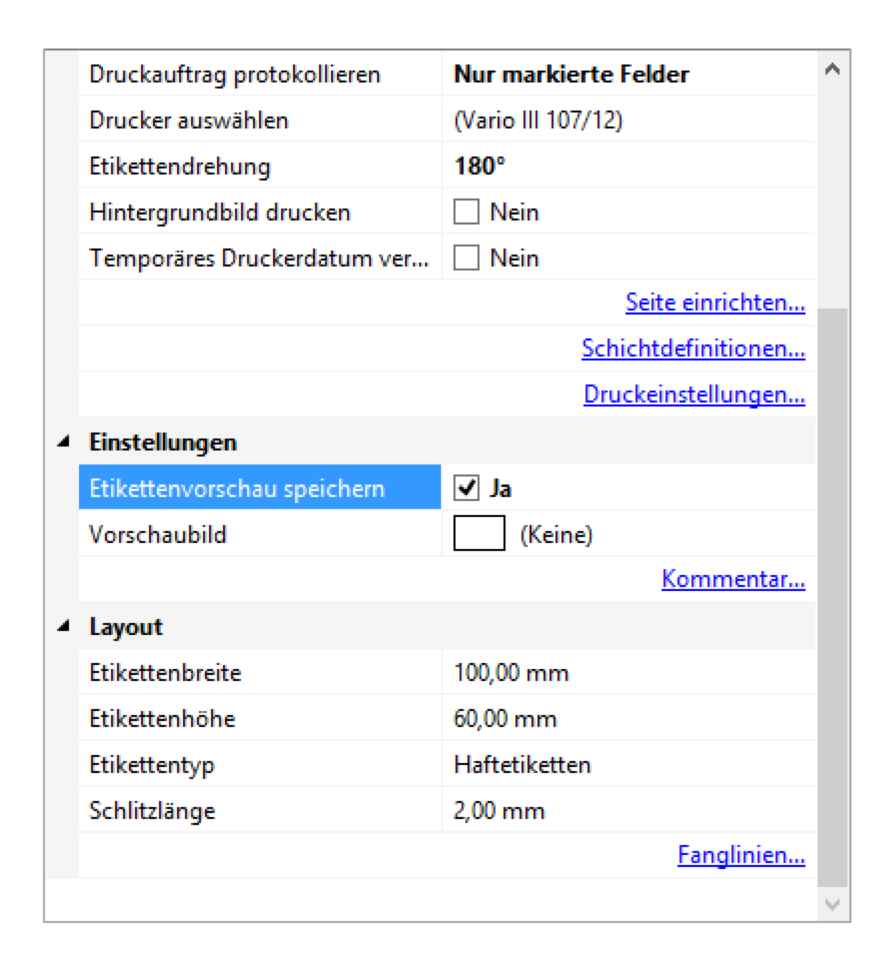
Der Standarddrucker, den Sie für **Labelstar Office** wählen, ist eine Programmeinstellung, d. h., alle Etiketten, die Sie mit **Labelstar Office** drucken, werden auf diesem Drucker ausgegeben, sofern Sie beim Etikett keinen anderen auswählen.

Registerkarte «Etikettenvorschau»

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

In dieser Registerkarte können Sie verschiedene Einstellungen (wie Größe, Ausgabeformat und Farbtiefe) des Vorschaubilds, das parallel zum Etikett gespeichert wird, festlegen.

Aktivieren Sie die Option **Etikettenvorschau für alle Etiketten speichern**, wenn Sie wollen dass zu jedem Etikett eine Vorschau gespeichert werden soll. Möchten Sie die Etikettenvorschau nur für bestimmte Etiketten aktivieren aktivieren Sie die Option **Etikettenvorschau speichern** in den Etiketteneinstellungen.



Registerkarte «Memory Card»

Benötigte Programmvariante	BASIC, PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

In dieser Registerkarte können Sie verschiedene Memory Card-Optionen ändern, z.B. das Standardverzeichnis für das Drucker- und das Systemlaufwerk.

Registerkarte «Protokollierung»

Benötigte Programmvariante	PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

In dieser Registerkarte können Sie die Protokolleinstellungen ändern.

Protokolldatei

Sie können angeben, wo die Protokolldatei gespeichert werden soll.

Verzeichnis Geben Sie den Namen des Ordners ein, in dem die Protokolldatei erstellt werden soll, oder klicken Sie auf [...], um den Ordner zu suchen.

Dateiname Geben Sie einen festen Dateinamen an oder verwenden Sie Platzhalter *%date%*, *%time%*, *%labelname%*, *%printername%*, die durch aktuelle Werte ersetzt werden, um einen variablen Dateinamen zu definieren. (z.B. *%labelname% %date%.log*).

Hinweis

Wählen Sie nur einen Pfad aus, auf den alle Benutzer Zugriff haben. Wählen Sie niemals nur *C:* oder *C:\Windows* aus. Wenn überhaupt, dann erstellen Sie bitte einen neuen Ordner und lassen das Programm dort seine Protokolldateien ablegen (z.B. *C:\Log*).

Vorhandene Protokolldatei überschreiben Aktivieren Sie dieses Kontrollkästchen, wenn die Protokolldatei die vorhandene Protokolldatei überschreiben und ersetzen soll.

Ortszeit für die Dateinamenerstellung verwenden Aktivieren Sie dieses Kontrollkästchen, wenn zum Erstellen des Protokolldateinamens die Ortszeit verwendet werden soll. Ist diese Option nicht ausgewählt, wird die koordinierte Weltzeit (Coordinated Universal Time/UTC) verwendet.

Protokolldateiformat

In diesem Abschnitt können Sie das Dateiformat definieren. Die Protokolldatei wird im CSV-Format gespeichert. Weitere Informationen zu den Daten, die gespeichert werden finden Sie unter Protokollierung.

Protokolldateirollover

In diesem Abschnitt legen Sie fest, ob und wann eine neue Protokolldatei begonnen werden soll.

Maximale Dateigröße (in MB) Wählen Sie diese Option, um das Erstellen zusätzlicher Protokolldateien zu ermöglichen, wenn die maximale Dateigröße erreicht wird. Jeder neue Dateiname besteht aus dem ursprünglichen Namen der Protokolldatei mit einer aufsteigenden Nummer.

Keine neue Protokolldatei erstellen Alle Informationen werden in einer einzigen, größer werdenden Datei protokolliert.

Registerkarte «Dateiablage»

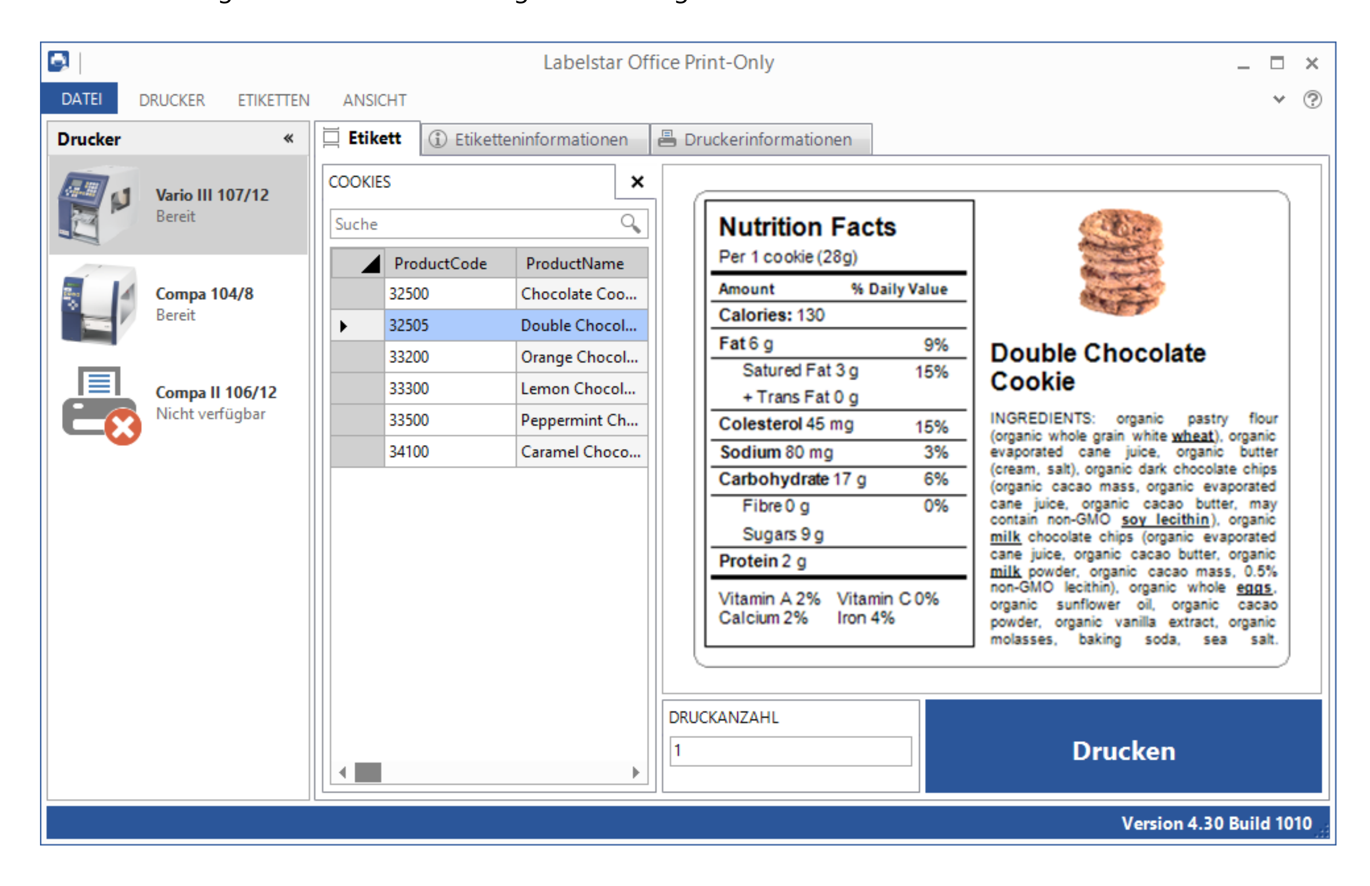
In dieser Registerkarte können Sie den Speicherort der im Programm verwendeten Verzeichnisse und Dateien ändern.

Um den Speicherort zu ändern, gehen Sie bitte folgendermaßen vor:

1. Wählen Sie einen Eintrag aus.
2. Klicken Sie auf ....
   Das Dialogfeld **Speicherort ändern** wird geöffnet.
3. Wählen Sie einen neuen Speicherort aus.

Print-Only

Mit diesem Programm können Etiketten geöffnet und gedruckt werden.



Tools

Labelstar Office stellt verschiedene Zusatzprogramme zur Verfügung, mit deren Hilfe die Programmdaten verwaltet und angepasst werden können.

Installierte Zusatzprogramme



Lizenzierungs-Assistent
Mit dieser Anwendung können Sie **Labelstar Office** lizenzieren. Weitere Informationen finden Sie unter Lizenzierung.



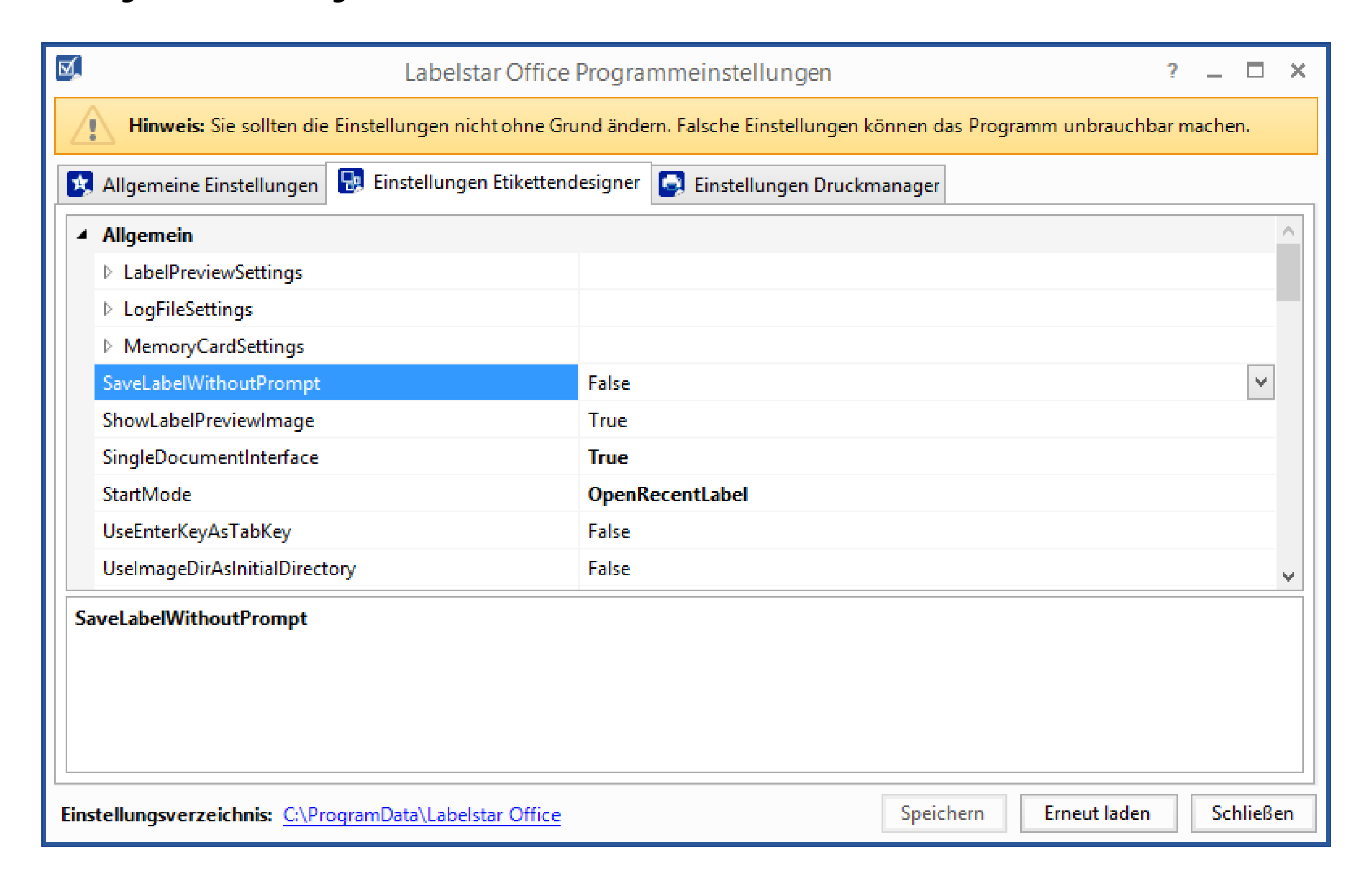
Programmeinstellungen
Mit dieser Anwendung können die internen Programmeinstellungen geändert werden.



Spracheinstellungen
Mit dieser Anwendung können Sie die Sprache auswählen, die für die Benutzeroberfläche (zum Beispiel in den Menüs, Dialogen und Hilfedateien) verwendet werden soll.

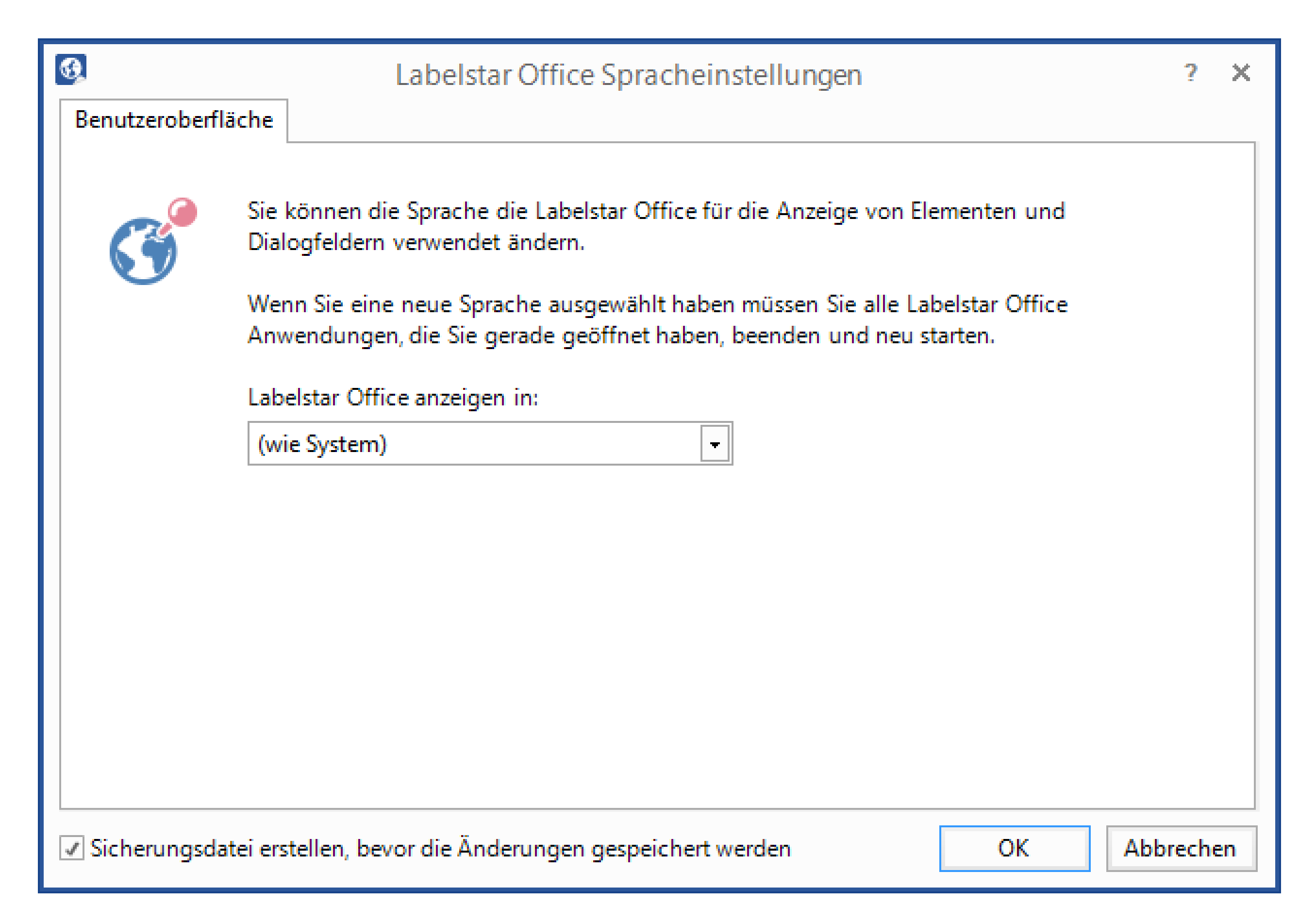
Programmeinstellungen

- Mit dieser Anwendung können die internen Programmeinstellungen geändert werden.
- Standardmäßig werden die Programmeinstellungen im Verzeichnis `C:\ProgramData\Labelstar Office` gespeichert.
- Um das Programm aufzurufen, klicken Sie auf **Start**, **Alle Programme**, **Labelstar Office**, **Tools** und dann auf **Programmeinstellungen**.



Spracheinstellungen

- Mithilfe des des Spracheinstellungstools können Sie die Spracheinstellungen von **Labelstar Office** ändern.
- Um das Programm aufzurufen, klicken Sie auf **Start**, **Alle Programme**, **Labelstar Office**, **Tools** und dann auf **Spracheinstellungen**.



Hinweis

Um die neuen Einstellungen zu übernehmen, müssen Sie alle **Labelstar Office** Anwendungen, die Sie gerade geöffnet haben, beenden und neu starten.

OLE-Automation

Benötigte Programmvariante	PROFESSIONAL
Weitere Informationen finden Sie unter Programmvarianten.	

Die **OLE-Automation** stellt einen Weg zur Verfügung damit OLE-konforme Anwendungen mit **Labelstar Office** interagieren können. Über OLE können anderen Programme eine **Labelstar Office** Sitzung starten, ein Etikett öffnen, den Etiketteninhalt ändern, das Etikett drucken und speichern.

Kurzanleitung

Application-Klasse
Repräsentiert die **Labelstar Office** Anwendung. Von diesem Objekt werden alle anderen Objekte abgeleitet.

Label-Klasse
Bietet Zugriff auf ein einzelnes Etikett.

Field-Klasse
Bietet Zugriff auf ein einzelnes Feld auf dem Etikett.



Hinweis

Auf ihrem Entwicklungscomputer muss eine Kopie der OLE-Automation-Anwendung vorhanden sein bevor Sie ihre Anwendung testen können. Ihr Anwender muss ebenfalls eine Kopie der OLE-Automation-Anwendung haben. Ohne **Labelstar Office** können Sie OLE-Automation nicht nutzen um Etiketten zu öffnen und zu drucken.

Was ist OLE-Automation?

OLE-Automation ist ein Set von Software Standards, die es einem Windows Programm erlauben, Objekte zu kontrollieren, die von anderen Windows-Anwendungen erstellt worden sind. Sobald man ein OLE-Automation-Objekt erstellt hat, kann man seine Eigenschaften ändern und Methoden aufrufen.

Was ist COM?

Bei dem **Component-Object-Model** (COM) von Microsoft handelt sich um modulare, objektorientierte Softwarekomponenten. Diese können von jeder COM-kompatiblen Anwendung genutzt werden. Mit dieser Technologie soll dem Microsoft-Betriebssystem Windows ermöglicht werden, Interprozesskommunikation und dynamische Objekterzeugung zu betreiben. In COM-kompatiblen Anwendungen können COM-Komponenten eingefügt und auch während der Laufzeit wieder entfernt werden. COM-Objekte können lauffähige Programme sein oder eine Programmbibliothek (Dynamic Link Library/DLL). Das **Component-Object-Model** ermöglicht dem Entwickler sprachunabhängig, plattformunabhängig, objektorientiert, versionsunabhängig, automatisiert und ortsunabhängig zu programmieren. Folgende Technologieren basieren auf COM-Komponenten: DirectX, ActiveX and OLE.

Systemanforderungen

Die OLE-Automation-Schnittstelle von **Labelstar Office** läuft auf Microsoft Windows Betriebssystemen (Windows 7/8/8.1 x86/x64).

Labelstar Office benötigt **.NET Framework 4.0 or higher**. Weitere Informationen und Download Links finden sie unter http://www.microsoft.com/net/.

Labelstar Office ist unter Verwendung von **x86 Platform target** kompiliert worden. Das heißt, dass das Projekt nur als 32-Bit-Prozess ausgeführt werden kann. Ein 64-Bit-Prozess kann kein 32-Bit-Assembly aufrufen. 32-Bit-Anwendungen und -Assemblies können auf Windows 64-Bit installiert werden. Sie werden jedoch unter WOW64 ausgeführt.

Assembly registrieren

Labelstar Office beinhaltet eine OLE-Automation-Schnittstelle die wie eine COM-Komponente mit IntelliSense-Unterstützung z.B. in Visual Basic 6, html pages, Delphi und Visual FoxPro verwendet werden kann.

Das Assembly wird automatisch registriert wenn Sie **Labelstar Office** auf ihrem Computer installieren.

LSOffice.dll befindet sich im Installationsverzeichnis und kann auf anderen Computer, unter Verwendung des Befehls *regasm* mit Administator-Rechten, registiert werden:

%SystemRoot%\Microsoft.NET\Framework\v4.0.30319\regasm.exe LSOffice.dll /codebase

%SystemRoot% ist das Windowsverzeichnis, normalerweise *C:\Windows* oder *C:\WINNT*. Das obige Beispiel geht davon aus, dass *LSOffice.dll* im aktuellen Arbeitsverzeichnis abgelegt ist. Andernfalls müssen Sie den absoluten Pfad zu *LSOffice.dll* angeben.


Hinweis

Wenn Sie das Assembly einsetzen wollen, stellen Sie sicher, dass **Labelstar Office** zusammen mit dem Assembly auf dem Zielsystem installiert ist.

Erste Schritte

Öffnen Sie die 32-Bit Version des Visual Basic Script Editors.



Hinweis

Labelstar Office wird unter Verwendung von **x86 Platform target** kompiliert. Das heißt, dass das Projekt nur als 32-Bit Prozess laufen kann.

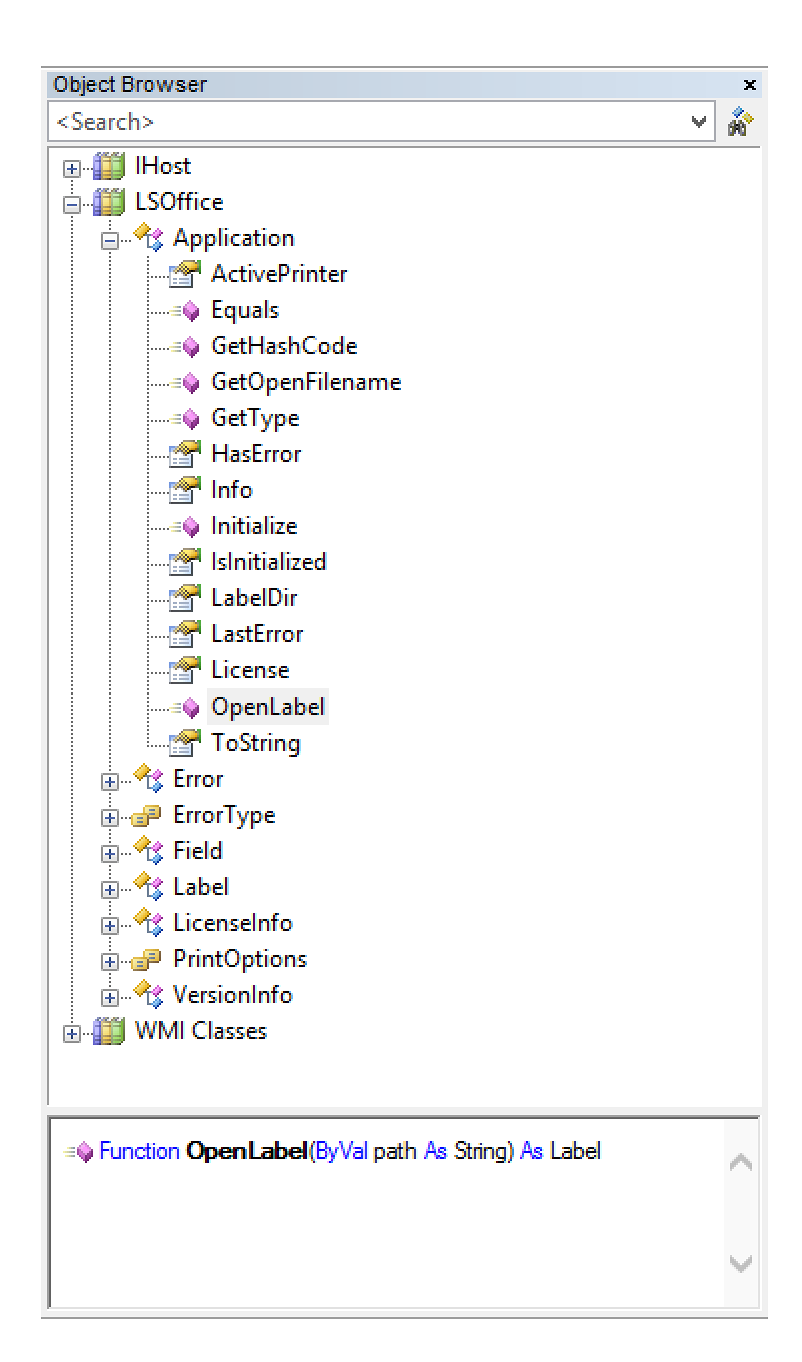
Referenzierung der Labelstar Office Type Library

Um **Labelstar Office** in einem Visual Basic Projekt zu verwenden, müssen Sie dem Projekt zuerst eine Referenz zur **Labelstar Office** Type Library hinzufügen.

Um die **Labelstar Office** Type Library zu referenzieren, gehen Sie wie folgt vor:

1. Klicken Sie in der Menüleiste auf **Extras** und wählen Sie dann **Referenzen** aus.
2. Falls die Schnittstelle bereits aufgelistet ist, wählen Sie diese aus ansonsten klicken Sie auf **Hinzufügen/Suchen**.
3. Wählen Sie die Datei LSOffice.tlb aus die im **Labelstar Office** Installationsverzeichnis abgelegt ist und klicken Sie dann auf **Öffnen**.

Jetzt können Sie sich die LSOffice-Objekte, ihre Methoden, Eigenschaften, Parameter und Aufzählungstypen unter Verwendung des Visual Basic Objekt Browsers anzeigen lassen.



Labelstar Office Runtime starten

Zuerst müssen Sie ein LSOffice.Application-Objekt erzeugen.

```
Dim objApp
objApp = CreateObject ("LSOffice.Application")
```

Danach müssen Sie die **Labelstar Office** Runtime initialisieren.

```
objApp.Initialize()
```

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Open And Print Label Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Open and print label
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Browse file name
Dim fileName
fileName = objApp.GetOpenFilename("Labels|*.lbex|All Files|*.*")

If (Len(fileName) = 0) Then
     WScript.Quit
End If

' Open label
Set objLabel = objApp.OpenLabel(fileName)

If (objLabel is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Print label
objLabel.Print(1)
```

Beispiele (VBScript)

Hinweis

Labelstar Office wird unter Verwendung von **x86 Platform target** kompiliert. Der Betrieb von **Labelstar Office** in einer 64-Bit Umgebung erforderte einen 32-Bit Script Host der im Ordner **SYSWOW64** abgelegt ist. Standardmäßig startet Windows 64-Bit die 64-Bit Version von wscript.exe (VBS Interpret). Das führt zur Fehlermeldung "800a01ad Active X Komponente kann das Objekt nicht erstellen".

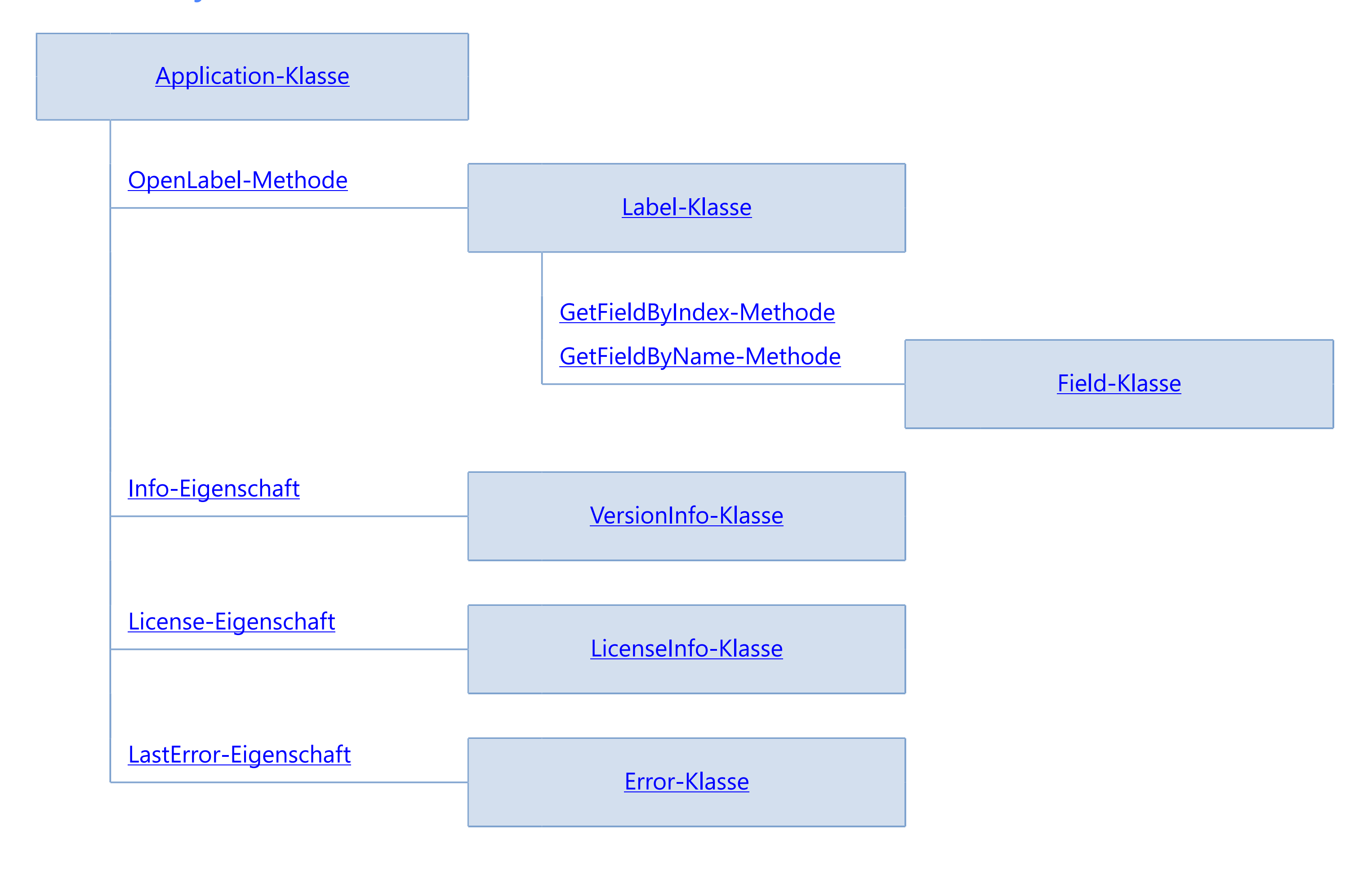
Die Beispielscripte zeigen wie man Etiketten öffnet, ändert und druckt. Die Scripte finden Sie im folgenden Verzeichnis: *%InstallDir%\Samples\COM Interop\VBScript*.

Beispielscript	Beschreibung
Open and print label.vbs	Zeigt ein Dialogfenster um ein Etikett auszuwählen, zu öffnen und zu drucken. Application.GetOpenFilename-Methode Application.OpenLabel-Methode Label.Print-Methode
Change printer name.vbs	Öffnet das Etikett ..*label1.lbex* und zeigt ein Dialogfenster zur Auswahl des aktiven Druckers. Label.ActivePrinter-Eigenschaft
Change field content.vbs	Öffnet das Etikett ..*label1.lbex* und zeigt ein Dialogfenster um den Feldinhalt von *Text1* zu ändern. Label.GetFieldByName-Methode Field.GetContent-Methode Field.SetContent-Methode
Set printable property.vbs	Öffnet das Etikett ..*label1.lbex* und zeigt ein Dialogfenster um auszuwählen, ob das Feld *Barcode1* gedruckt werden soll oder nicht. Field.Printable-Eigenschaft
Change text alignment.vbs	Öffnet das Etikett ..*label4.lbex* und zeigt ein Dialogfenster um die Textausrichtung des Felds *Text1* auszuwählen. Field.SetPropertyValue-Methode
Display field names.vbs	Zeigt ein Dialogfenster mit den definierten Feldern von ..*label1.lbex*, ..*label2.lbex* und ..*label3.lbex*. Label.FieldNames-Eigenschaft
Display last error.vbs	Zeigt die Fehlerbehandlung. Application.LastError-Eigenschaft
Print record.vbs	Öffnet das Datenbanketikett ..*label3.lbex* und zeigt ein Dialogfenster um einen Suchstring einzugeben. **Hinweis:** Die Datenverbindung *Europe* muss in **Labelstar Office** definiert sein. Label.SelectRecord-Methode

Objektmodellreferenz

Die API-Schnittstelle von **Labelstar Office** stellt Klassen, Schnittstellen und Werttypen zur Verfügung, die in LSOffice.dll enthalten sind. Diese Eigenschaften und Methoden können in Kundenanwendungen, zur Steuerung des Etikettendrucks, verwendet werden.

LSOffice Objekthierarchie



Application-Klasse

Ein **Application**-Objekt repräsentiert die gesamte **Labelstar Office** Anwendung und ist das oberste Objekt der Objekthierarchie, von dem aus alle anderen Objekte abgeleitet werden.

So können Sie ein **Application**-Objekt erstellen:

```
Dim objApp
Set objApp = CreateObject("LSOffice.Application")

objApp.Initialize()
```

Eigenschaften

	Name	Beschreibung
	ActivePrinter	Gibt den Namen des aktiven Druckers zurück.
	HasError	Ruft einen Wert ab, der angibt, ob während des letzten Aufrufs einer Methode oder Eigenschaft ein Fehler aufgetreten ist.
	Info	Ruft ein Objekt ab, das Eigenschaften zum Abrufen von Informationen über die API bereitstellt, z. B. Versionsnummer, Copyright usw.
	IsInitialized	Ruft einen Wert ab, der angibt, ob die Initialize-Methode aufgerufen worden ist oder nicht.
	LabelDir	Gibt den Pfad des aktuellen Etikettenverzeichnisses zurück.
	LastError	Ruft den letzten Fehler ab.
	License	Ruft ein Objekt ab, das Eigenschaften zum Abrufen der Lizenzinformationen bereitstellt.

Methoden

	Name	Beschreibung
	Initialize	Initialisiert das Application-Objekt auf Grundlage der aktuellen Programmeinstellungen.
	GetOpenFilename	Zeigt das standardmäßige Dialogfeld **Öffnen** an und ruft einen Dateinamen vom Benutzer ab, ohne Dateien versehentlich zu öffnen.
	OpenLabel	Öffnet das angegebene Etikett.

Siehe auch

- Objektmodellreferenz

Application-Eigenschaften

Dieses Objekt hat die folgenden Eigenschaften.

Eigenschaften

	Name	Beschreibung
	ActivePrinter	Gibt den Namen des aktiven Druckers zurück.
	HasError	Ruft einen Wert ab, der angibt, ob während des letzten Aufrufs einer Methode oder Eigenschaft ein Fehler aufgetreten ist.
	Info	Ruft ein Objekt ab, das Eigenschaften zum Abrufen von Informationen über die API bereitstellt, z. B. Versionsnummer, Copyright usw.
	IsInitialized	Ruft einen Wert ab, der angibt, ob die Initialize-Methode aufgerufen worden ist oder nicht.
	LabelDir	Gibt den Pfad des aktuellen Etikettenverzeichnisses zurück.
	LastError	Ruft den letzten Fehler ab.
	License	Ruft ein Objekt ab, das Eigenschaften zum Abrufen der Lizenzinformationen bereitstellt.

Siehe auch

- Application-Klasse
- Objektmodellreferenz

ActivePrinter-Eigenschaft

Gibt den Namen des aktiven Druckers zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.ActivePrinter
```

Typ

```
String
```

Beispiel (VBScript)

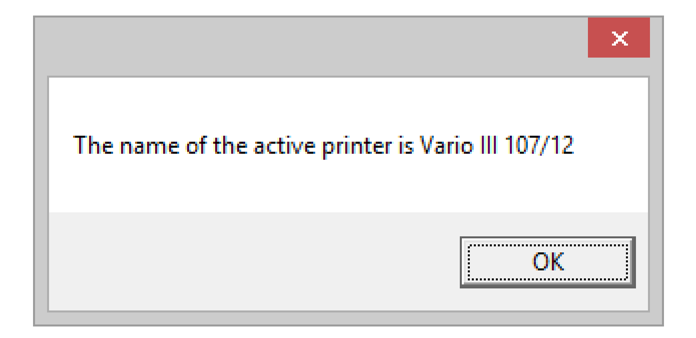
Im folgenden Beispiel wird der Name des aktiven Druckers angezeigt.

```
Dim objApp
Set objApp = CreateObject("LSOffice.Application")

objApp.Initialize()

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

MsgBox "The name of the active printer is " & objApp.ActivePrinter
```



Siehe auch

- Application-Klasse
- Objektmodellreferenz

HasError-Eigenschaft

Ruft einen Wert ab, der angibt, ob während des letzten Aufrufs einer Methode oder Eigenschaft ein Fehler aufgetreten ist. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.HasError
```

Typ

```
Boolean
```

Hinweis

Weitere Informationen finden Sie unter Application.LastError.

Siehe auch

- Application-Klasse
- Objektmodellreferenz

Info-Eigenschaft

Ruft ein Objekt ab, das Eigenschaften zum Abrufen von Informationen über die API bereitstellt, z. B. Versionsnummer, Copyright usw. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.Version
```

Typ

`LSOffice.VersionInfo`

Siehe auch

- VersionInfo-Klasse
- Application-Klasse
- Objektmodellreferenz

IsInitialized-Eigenschaft

Ruft einen Wert ab, der angibt, ob die Initialize-Methode aufgerufen worden ist oder nicht. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.IsInitialized
```

Typ

```
Boolean
```

Hinweis

Verwenden Sie diese Eigenschaft, um zu prüfen ob das Application-Objekt bereits initialisiert wurde. Die Initialize-Methode sollte nur einmal aufgerufen werden.

Siehe auch

- Application-Klasse
- Objektmodellreferenz

LabelDir-Eigenschaft

Gibt den Pfad des aktuellen Etikettenverzeichnisses zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.LabelDir
```

Typ

```
String
```

Siehe auch

- Application-Klasse
- Objektmodellreferenz

LastError-Eigenschaft

Ruft den letzten Fehler ab. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Usage

`objApp.LastError`

Type

LSOffice.Error

Hinweis

Diese Eigenschaft enthält einen Fehlercode, der den Grund für die fehlerhafte Operation angibt. Überprüfen Sie Application.HasError um festzustellen, ob eine Operation fehlgeschlagen ist oder nicht. Weitere Informationen finden Sie unter Fehlercodes.

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Display Last Error Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Constants LSOffice.ErrorType
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Const ErrorType_Success = 0
Const ErrorType_Warning = 1
Const ErrorType_Error = 2

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' DisplayLastError
' Purpose:
'    Shows a message box displaying the last error.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Sub DisplayLastError()

     If (objApp.LastError.ErrorType = ErrorType_Success) Then
          Exit Sub
     End If

     Dim title
```

```
    title = "Error"

    If (objApp.LastError.ErrorType = ErrorType_Warning) Then
        title = "Message"
    End If

    MsgBox objApp.LastError.Message, vbOKOnly, title

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Open and print label
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()
DisplayLastError()

If (objApp.HasError) Then
    WScript.Quit
End If

' Open label
Set objLabel = objApp.OpenLabel("..\Label10.lbex")

If (objLabel is Nothing) Then
    DisplayLastError()
    WScript.Quit
End If
```

Siehe auch

- Error-Klasse
- Application-Klasse
- Objektmodellreferenz

License-Eigenschaft

Ruft ein Objekt ab, das Eigenschaften zum Abrufen der Lizenzinformationen bereitstellt. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.License
```

Typ

`LSOffice.LicenseInfo`

Hinweis

In der Testversion wird zu jeder Grafik ein Wasserzeichen hinzugefügt und es werden alle 'e' durch 'x' und alle '5' durch '0' ersetzt.

Siehe auch

- LicenseInfo-Klasse
- Application-Klasse
- Objektmodellreferenz

Application-Methoden

Dieses Objekt hat die folgenden Methoden.

Methoden

	Name	Beschreibung
	Initialize	Initialisiert das Application-Objekt auf Grundlage der aktuellen Programmeinstellungen.
	GetOpenFilename	Zeigt das standardmäßige Dialogfeld **Öffnen** an und ruft einen Dateinamen vom Benutzer ab, ohne Dateien versehentlich zu öffnen.
	OpenLabel	Öffnet das angegebene Etikett.

Siehe auch

- Application-Klasse
- Objektmodellreferenz

Initialize-Methode

Initialisiert das Application-Objekt auf Grundlage der aktuellen Programmeinstellungen.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.Initialize ()
```

Hinweis

Bevor ein Application-Objekt verwendet werden kann muss es mit dieser Methode initialisiert werden. Überprüfen Sie die IsInitialized-Eigenschaft, um festzustellen ob die Funktion bereits aufgerufen worden ist oder nicht.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter OLE-Automation -> Erste Schritte.

Siehe auch

- Application-Klasse
- Objektmodellreferenz

GetOpenFilename-Methode

Zeigt das standardmäßige Dialogfeld **Öffnen** an und ruft einen Dateinamen vom Benutzer ab, ohne Dateien versehentlich zu öffnen.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.GetOpenFilename (filter, [filterIndex], [title])
```

Parameter

filter

Typ: `String`
Eine Zeichenfolge, die Dateifilterkriterien angibt.

Die Zeichenfolge besteht aus Dateifilter-Zeichenfolgenpaaren, wobei jedes Paar aus einer Dateifilterbezeichnung gefolgt von der MS-DOS-Platzhalter-Dateifilterspezifikation besteht. Jeder Teil und jedes Paar sind durch einen vertikalen Strich (|) voneinander getrennt. Beispielsweise gibt die folgende Zeichenfolge zwei Dateifilter an: „Etiketten|*.lbex|Alle Dateien|*.*".

Trennen Sie zum Verwenden mehrerer MS-DOS-Platzhalterausdrücke für einen einzelnen Dateifiltertyp die Platzhalterausdrücke mit Semikolons, beispielsweise „Visual Basic-Dateien|*.bas;*.txt".

filterIndex (optional)

Typ: `Integer`
Gibt an welches Dateifilterkriterium, und zwar von 1 zur Anzahl der in *filter* angegebenen Filter, beim Öffnen des Dialogfensters angezeigt werden soll. Ist dieses Argument nicht definiert oder größer als die Anzahl der definierten Filter, wird standardmäßig der erste Filter angezeigt.

title (optional)

Typ: `String`
Gibt den Titel des Dialogfelds an. Ist dieses Argument nicht definiert, wird standardmäßig "Öffnen" als Title verwendet.

Rückgabetyp

`String`

Hinweis

Diese Methode gibt den im Dialogfeld ausgewählten Dateinamen oder den vom Benutzer eingegebenen Namen zurück, oder eine leere Zeichenfolge (""), wenn der Benutzer das Dialogfenster abbricht. Der zurückgegebene Name enthält möglicherweise eine Pfadspezifikation.

Diese Methode ändert möglicherweise das aktuelle Laufwerk bzw. den aktuellen Ordner.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter OLE-Automation -> Erste Schritte.

Siehe auch

- Application-Klasse

Objektmodellreferenz

OpenLabel-Methode

Öffnet das angegebene Etikett.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objApp.OpenLabel (path)
```

Parameter

path
Typ: `String`
Der Pfad des Etiketts.

Rückgabetyp

`LSOffice.Label`

Hinweis

Gibt ein Label-Objekt zurück wenn das Etikett erfolgreich geöffnet worden ist, oder **null**, wenn beim Öffnen des Etiketts ein Fehler aufgetreten ist.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter OLE-Automation -> Erste Schritte.

Siehe auch

- Label-Klasse
- Application-Klasse
- Objektmodellreferenz

Error-Klasse

Stellt Eigenschaften zum Abrufen des aktuellen Fehlerstatus bereit.

Eigenschaften

	Name	Beschreibung
	Details	Detaillierte Fehlerbeschreibung des zuletzt aufgetretenen Fehlers.
	ErrorCode	Fehlercode des zuletzt aufgetretenen Fehlers.
	ErrorType	Gibt den Fehlertyp zurück.
	Message	Fehlerbeschreibung des zuletzt aufgetretenen Fehlers.

Siehe auch

- Objektmodellreferenz

Error-Eigenschaften

Dieses Objekt hat die folgenden Eigenschaften.

Eigenschaften

	Name	Beschreibung
	Details	Detaillierte Fehlerbeschreibung des zuletzt aufgetretenen Fehlers.
	ErrorCode	Fehlercode des zuletzt aufgetretenen Fehlers.
	ErrorType	Gibt den Fehlertyp zurück.
	Message	Fehlerbeschreibung des zuletzt aufgetretenen Fehlers.

Siehe auch

- Error-Klasse
- Objektmodellreferenz

Details-Eigenschaft

Detaillierte Fehlerbeschreibung des zuletzt aufgetretenen Fehlers. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objError.Details
```

Typ

```
String
```

Hinweis

Gibt eine detaillierte Fehlermeldung (incl. Stack Trace) zurück, die die Ursache des zuletzt aufgetretenen Fehlers erklärt, oder eine leere Zeichenfolge (""), wenn kein Fehler aufgetreten ist.

Siehe auch

- Error-Klasse
- Objektmodellreferenz

ErrorCode-Eigenschaft

Fehlercode des zuletzt aufgetretenen Fehlers. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objError.ErrorCode
```

Typ

```
Int32
```

Hinweis

Gibt den Fehlercode des zuletzt aufgetretenen Fehlers zurück, oder 0, wenn kein Fehler aufgetreten ist. Weitere Informationen finden Sie unter Fehlercodes.

Siehe auch

- Error-Klasse
- Objektmodellreferenz

ErrorType-Eigenschaft

Gibt den Fehlertyp zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objError.ErrorType
```

Typ

`LSOffice.ErrorType`

Hinweis

Gibt den Fehlertyp des zuletzt aufgetretenen Fehlers zurück, oder Success, wenn kein Fehler aufgetreten ist.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Application.LastError.

Siehe auch

- Error-Klasse
- ErrorType-Enumeration
- Objektmodellreferenz

Message-Eigenschaft

Fehlerbeschreibung des zuletzt aufgetretenen Fehlers. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objError.Message
```

Typ

```
String
```

Hinweis

Gibt eine Fehlermeldung zurück, die die Ursache des zuletzt aufgetretenen Fehlers erklärt, oder eine leere Zeichenfolge (""), wenn kein Fehler aufgetreten ist. Weiter Informationen finden Sie unter Fehlercodes.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Application.LastError.

Siehe auch

- Error-Klasse
- Objektmodellreferenz

ErrorType-Enumeration

Gibt die verschiedenen Fehlertypen an.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Member

	Name	Wert	Beschreibung
	Success	0 (0x00)	Es ist kein Fehler aufgetreten.
	Warning	1 (0x01)	Der Aufruf war erfolgreich, aber es gibt ein mögliches Problem.
	Error	2 (0x02)	Der Aufruf ist fehlgeschlagen.

Hinweis

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Application.LastError.

Siehe auch

- Error-Klasse
- Objektmodellreferenz

Field-Klasse

Ein **Field**-Objekt stellt ein Feld auf einem Etikett dar. Es kann verwendet werden, um den Feldinhalt bzw. -eigenschaften auszulesen und zu ändern. Sie können eine Referenz auf ein **Field**-Object über die Methode Label.GetFieldByIndex oder Label.GetFieldByName erstellen.

Eigenschaften

	Name	Beschreibung
	FieldName	Gibt den Feldnamen zurück.
	Locked	Gibt einen Wert zurück der angibt, ob das Feld geändert werden kann oder nicht.
	Printable	Gibt einen Wert zurück der angibt, ob das Feld gedruckt wird oder nicht.

Methoden

	Name	Beschreibung
	GetContent	Gibt den aktuellen Feldinhalt zurück.
	GetPropertyValue	Gibt den Wert der Eigenschaft zurück.
	SetContent	Setzt den aktuellen Feldinhalt.
	SetPropertyValue	Setzt den Wert der Eigenschaft.

Siehe auch

- Objektmodellreferenz

Field-Eigenschaften

Dieses Objekt hat die folgenden Eigenschaften.

Eigenschaften

	Name	Beschreibung
🔒	FieldName	Gibt den Feldnamen zurück.
🔒	Locked	Gibt einen Wert zurück der angibt, ob das Feld geändert werden kann oder nicht.
	Printable	Gibt einen Wert zurück der angibt, ob das Feld gedruckt wird oder nicht.

Siehe auch

- Field-Klasse
- Objektmodellreferenz

FieldName-Eigenschaft

Gibt den Feldnamen zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.FieldName
```

Typ

```
String
```

Siehe auch

- Field-Klasse
- Objektmodellreferenz

Locked-Eigenschaft

Gibt einen Wert zurück der angibt, ob das Feld geändert werden kann oder nicht. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.Locked
```

Typ

```
Boolean
```

Siehe auch

- Field-Klasse
- Objektmodellreferenz

Printable-Eigenschaft

Gibt einen Wert zurück der angibt, ob das Feld gedruckt wird oder nicht.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.Printable
```

Typ

```
Boolean
```

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Set Printable Property Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel
Dim objField

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Set printable property
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Open label
Set objLabel = objApp.OpenLabel("..\Label1.lbex")

If (objLabel is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Get field by name
Set objField = objLabel.GetFieldByName("Barcode1")
```

```
If (objField Is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Set property and print label
Dim msg
msg = MsgBox("Print barcode?", vbYesNo, objField.FieldName)

objField.Printable = (msg = vbYes)
objLabel.Print(1)
```

Siehe auch

- Field-Klasse
- Objektmodellreferenz

Field-Methoden

Dieses Objekt hat die folgenden Methoden.

Methoden

	Name	Beschreibung
	GetContent	Gibt den aktuellen Feldinhalt zurück.
	GetPropertyValue	Gibt den Wert der Eigenschaft zurück.
	SetContent	Setzt den aktuellen Feldinhalt.
	SetPropertyValue	Setzt den Wert der Eigenschaft.

Siehe auch

- Field-Klasse
- Objektmodellreferenz

GetContent-Methode

Gibt den aktuellen Feldinhalt zurück.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.GetContent ()
```

Rückgabetyp

```
String
```

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Change Field Content Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel
Dim objField

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Change field content
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Open label
Set objLabel = objApp.OpenLabel("..\Label1.lbex")

If (objLabel is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Get field by name
Set objField = objLabel.GetFieldByName("Text1")
```

```
If (objField Is Nothing) Then
     WScript.Echo objApp.LastError.Message
    WScript.Quit
End If

' Enter new field content
Dim result
result = InputBox("Field content:", objField.FieldName, objField.GetContent())

' Evaluate the user input
If result <> "" Then
     ' Set field content and print label
     objField.SetContent(result)
     objLabel.Print(1)
End If
```

Siehe auch

- Field-Klasse
- Objektmodellreferenz

GetPropertyValue-Methode

Gibt den Wert der Eigenschaft zurück.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.GetPropertyValue (propertyName)
```

Parameter

propertyName
Typ: `String`
Eigenschaftenname

Die folgende Tabelle beschreibt einige mögliche Eigenschaftennamen:

Eigenschaftenname	Beschreibung
Printable	Typ: `Boolean` falsch oder 0: Feld wird nicht gedruckt wahr oder 1: Feld wird gedruckt siehe auch: Printable-Eigenschaft
Locked	Typ: `Boolean` falsch oder 0: Feld ist nicht gesperrt wahr oder 1: Feld ist gesperrt siehe auch: Locked-Eigenschaft
TextAlignment	**Nur Text/Barcode** Typ: `Integer` 0: Links 1: Zentriert 2: Recht 3: Blocksatz
HumanReadable	**Nur Barcode** Typ: `Boolean` falsch oder 0: Klartext wird nicht angezeigt wahr oder 1: Klartext wird angezeigt

Rückgabetyp

`Object`

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Change Text Alignment Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel
Dim objField

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Select text alignment
' Purpose:
'    Displays a message box to select the text alignment option.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function SelectTextAlignment

     SelectTextAlignment = -1

     Dim text

     text = "Text alignment:" & vbCrLf & vbCrLf
     text = text & "0" & vbTab & "Left aligned" & vbCrLf
     text = text & "1" & vbTab & "Centered" & vbCrLf
     text = text & "2" & vbTab & "Right aligned"

     ' Show all available printers and allow a user selection
     Dim tmp
     tmp = InputBox(text, "Select text alignment", "0")

     If tmp = "" Then
         WScript.Echo "No user input, aborted"
         Exit Function
     End If

     tmp = CInt(tmp)

     If (tmp < 0) Or (tmp > 2) Then
         WScript.Echo "Wrong value, aborted"
         Exit Function
     End If

     ' Set text alignment
     SelectTextAlignment = tmp

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Change text alignment
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Open label
```

```
Set objLabel = objApp.OpenLabel("..\Label4.lbex")

If (objLabel is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Get field by name
Set objField = objLabel.GetFieldByName("Text1")

If (objField is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Select text alignment
Dim textAlignment
textAlignment = SelectTextAlignment()

If (textAlignment < 0) Then
     WScript.Quit
End If

' Set text alignment and print label
objField.SetPropertyValue "TextAlignment", textAlignment
objLabel.Print(1)
```

Siehe auch

- Field-Klasse
- Objektmodellreferenz

SetContent-Methode

Setzt den aktuellen Feldinhalt.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.SetContent (content)
```

Parameter

content
Typ: `String`
Neuer Feldinhalt.

Hinweis

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Field.GetContent.

Siehe auch

- Field-Klasse
- Objektmodellreferenz

SetPropertyValue-Methode

Setzt den Wert der Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objField.SetPropertyValue (propertyName, value)
```

Parameter

propertyName
Typ: `String`
Eigenschaftenname

Die folgende Tabelle beschreibt einige mögliche Eigenschaftennamen:

Eigenschaftenname	**Beschreibung**
Printable	Typ: `Boolean` falsch oder 0: Feld wird nicht gedruckt wahr oder 1: Feld wird gedruckt siehe auch: Printable-Eigenschaft
Locked	Typ: `Boolean` falsch oder 0: Feld ist nicht gesperrt wahr oder 1: Feld ist gesperrt siehe auch: Locked-Eigenschaft
TextAlignment	**Nur Text/Barcode** Typ: `Integer` 0: Links 1: Zentriert 2: Recht 3: Blocksatz
HumanReadable	**Nur Barcode** Typ: `Boolean` falsch oder 0: Klartext wird nicht angezeigt wahr oder 1: Klartext wird angezeigt

value
Typ: `Object`
Neuer Eigenschaftenwert.

Hinweis

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Field.GetPropertyValue.

Siehe auch

- Field-Klasse

- Objektmodellreferenz

ImageFormat-Enumeration

Liste der unterstützten Dateiformate.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Member

	Name	Wert	Beschreibung
	Bmp	0 (0x00)	Speichert die Grafik im BMP-Format.
	Gif	1 (0x01)	Speichert die Grafik im GIF-Format.
	Jpeg	2 (0x02)	Speichert die Grafik im JPEG-Format.
	Png	3 (0x03)	Speichert die Grafik im PNG-Format.

Siehe auch

- SavePreview-Methode
- Objektmodellreferenz

Label-Klasse

Ein **Label**-Objekt repräsentiert ein geöffnetes Etikett. Sie erhalten eine Referenz für ein **Label**-Objekt wenn Sie die Methode Application.OpenLabel aufrufen.

Eigenschaften

	Name	Beschreibung
	ActivePrinter	Gibt den Namen des aktiven Druckers zurück oder legt ihn fest.
	CurrentRecord	Gibt den aktuellen Datensatz zurück oder legt ihn fest.
🔒	FieldCount	Gibt die Anzahl der Felder zurück, die auf dem Etikett definiert sind.
🔒	FieldNames	Gibt eine Liste der Feldnamen zurück, die auf dem Etikett definiert sind.
🔒	IsDataAvailable	Ruft einen Wert ab, der angibt, ob Datenbankfelder auf dem Etikett definiert sind oder nicht.
🔒	LabelPath	Gibt den Pfad des geöffneten Etiketts zurück.
🔒	MaxRecord	Gibt die maximale Anzahl von Datensätzen zurück.
🔒	Modified	Ruf einen Wert ab, der angibt, ob sich der Etiketteninhalt geändert hat oder nicht.
	PageName	Gibt den Seitennamen zurück oder legt ihn fest.

Methoden

	Name	Beschreibung
	GetFieldByIndex	Sucht das Feld mit dem angegebenen Index.
	GetFieldByName	Sucht das Feld mit dem angegebenen Namen.
	GetPreview	Erstellt ein Vorschaubild des aktuellen Etiketteninhalts.
	GetPropertyValue	Gibt den Wert der Eigenschaft zurück.
	Print	Druckt das Etikett.
	PrintToFile	Druck in Datei.
	Save	Speichert Änderungen am Etikett.
	SaveAs	Speichert Änderungen am Etikett unter einem anderen Namen.
	SavePreview	Speichert ein Vorschaubild des aktuellen Etiketteninhalts.
	SelectRecord	Sucht den angegebenen Datensatz.
	SetPropertyValue	Setzt den Wert der Eigenschaft zurück.

Siehe auch

- Objektmodellreferenz

Label-Eigenschaften

Dieses Objekt hat die folgenden Eigenschaften.

Eigenschaften

	Name	Beschreibung
	ActivePrinter	Gibt den Namen des aktiven Druckers zurück oder legt ihn fest.
	CurrentRecord	Gibt den aktuellen Datensatz zurück oder legt ihn fest.
	FieldCount	Gibt die Anzahl der Felder zurück, die auf dem Etikett definiert sind.
	FieldNames	Gibt eine Liste der Feldnamen zurück, die auf dem Etikett definiert sind.
	IsDataAvailable	Ruft einen Wert ab, der angibt, ob Datenbankfelder auf dem Etikett definiert sind oder nicht.
	LabelPath	Gibt den Pfad des geöffneten Etiketts zurück.
	MaxRecord	Gibt die maximale Anzahl von Datensätzen zurück.
	Modified	Ruf einen Wert ab, der angibt, ob sich der Etiketteninhalt geändert hat oder nicht.
	PageName	Gibt den Seitennamen zurück oder legt ihn fest.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

ActivePrinter-Eigenschaft

Gibt den Namen des aktiven Druckers zurück oder legt ihn fest.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.ActivePrinter
```

Typ

```
String
```

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Change Printer Name Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Select printer
' Purpose:
'    Displays a message box to select one of the available printers.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function SelectPrinter(activePrinter)

     SelectPrinter = ""

     ' Read all printers
     Dim wshNetwork, objPrinters
     Set wshNetwork = WScript.CreateObject("WScript.Network")
     Set objPrinters = wshNetwork.EnumPrinterConnections

     Dim text, i, j, index

     text = "Available printers:" & vbCrLf & vbCrLf
     j = objPrinters.Count
     index = 0

     For i = 0 To j - 1 Step 2

          If (objPrinters(i+1) = activePrinter) Then
               index = i/2
          End If
```

```
        text = text & (i/2) & vbTab
        text = text & objPrinters(i+1) & vbCrLf

    Next

    ' Show all available printers and allow a user selection
    Dim tmp
    tmp = InputBox(text, "Select printer", index)

    If tmp = "" Then
        WScript.Echo "No user input, aborted"
        Exit Function
    End If

    tmp = CInt(tmp)

    If (tmp < 0) Or (tmp > (j/2 - 1)) Then
        WScript.Echo "Wrong value, aborted"
        Exit Function
    End If

    ' Set printer name
    SelectPrinter = objPrinters(tmp*2 + 1)

End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Change printer name
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
    WScript.Echo objApp.LastError.Message
    WScript.Quit
End If

' Open label
Set objLabel = objApp.OpenLabel("..\Label1.lbex")

If (objLabel is Nothing) Then
    WScript.Echo objApp.LastError.Message
    WScript.Quit
End If

' Select printer
Dim activePrinter
activePrinter = SelectPrinter(objLabel.ActivePrinter)

If (activePrinter = "") Then
    WScript.Quit
End If

' Set active printer and print label
```

```
objLabel.ActivePrinter = activePrinter
objLabel.Print(1)
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

CurrentRecord-Eigenschaft

Gibt den aktuellen Datensatz zurück oder legt ihn fest.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.CurrentRecord
```

Typ

```
Integer
```

Hinweis

Mit der Eigenschaft IsDataAvailable können Sie überprüfen, ob Datenbankfelder auf dem Etikett definiert sind.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

FieldCount-Eigenschaft

Gibt die Anzahl der Felder zurück, die auf dem Etikett definiert sind. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.FieldCount
```

Typ

```
Integer
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

FieldNames-Eigenschaft

Gibt eine Liste der Feldnamen zurück, die auf dem Etikett definiert sind. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.FieldNames
```

Typ

```
String[]
```

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Display Field Names Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' DisplayFieldNames
' Purpose:
'    A message box is displayed showing all fields defined on the
'    label.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Sub DisplayFieldNames(labelName)

     ' Open label
     Dim objLabel
     Set objLabel = objApp.OpenLabel(labelName)

     If (objLabel is Nothing) Then
          WScript.Echo objApp.LastError.Message
          Exit Sub
     End If

     ' Format field names
     Dim fieldNames
     fieldNames = objLabel.FieldNames

     Dim text, i, j

     j = UBound(fieldNames)

     text = "Available fields:" & vbCrLf & vbCrLf
```

```
    For i = 0 To j Step 1
        text = text & i & vbTab
        text = text & fieldNames(i) & vbCrLf
    Next

    MsgBox labelName & vbCrLf & vbCrLf & text, vbOKOnly, "Field names"

End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Display field names
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
    WScript.Echo objApp.LastError.Message
    WScript.Quit
End If

' Display field names
DisplayFieldNames "..\Label1.lbex"
DisplayFieldNames "..\Label2.lbex"
DisplayFieldNames "..\Label3.lbex"
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

IsDataAvailable-Eigenschaft

Ruft einen Wert ab, der angibt, ob Datenbankfelder auf dem Etikett definiert sind oder nicht. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.IsDataAvailable
```

Typ

```
Boolean
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

LabelPath-Eigenschaft

Gibt den Pfad des geöffneten Etiketts zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.LabelPath
```

Typ

```
String
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

MaxRecord-Eigenschaft

Gibt die maximale Anzahl von Datensätzen zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.MaxRecord
```

Typ

```
Integer
```

Hinweis

Gibt die maximale Anzahl von Datensätzen in der Datenbank zurück, oder 0, wenn keine Datenbankfelder auf dem Etikett definiert sind. Mit der Eigenschaft IsDataAvailable können Sie überprüfen, ob Datenbankfelder auf dem Etikett definiert sind.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

Modified-Eigenschaft

Ruf einen Wert ab, der angibt, ob sich der Etiketteninhalt geändert hat oder nicht.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.Modified
```

Typ

```
Boolean
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

PageName-Eigenschaft

Gibt den Seitennamen zurück oder legt ihn fest.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Verwendung

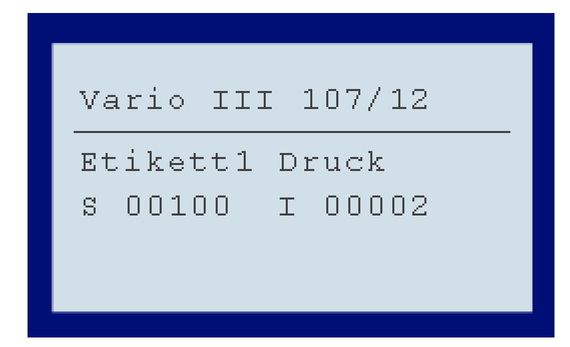
```
objLabel.PageName
```

Typ

```
String
```

Hinweis

Der Seitenname wird während des Drucks im Druckerdisplay angezeigt.



Siehe auch

- Label-Klasse
- Objektmodellreferenz

Label-Methoden

Dieses Objekt hat die folgenden Methoden.

Methoden

	Name	Beschreibung
	GetFieldByIndex	Sucht das Feld mit dem angegebenen Index.
	GetFieldByName	Sucht das Feld mit dem angegebenen Namen.
	GetPreview	Erstellt ein Vorschaubild des aktuellen Etiketteninhalts.
	GetPropertyValue	Gibt den Wert der Eigenschaft zurück.
	Print	Druckt das Etikett.
	PrintToFile	Druck in Datei.
	Save	Speichert Änderungen am Etikett.
	SaveAs	Speichert Änderungen am Etikett unter einem anderen Namen.
	SavePreview	Speichert ein Vorschaubild des aktuellen Etiketteninhalts.
	SelectRecord	Sucht den angegebenen Datensatz.
	SetPropertyValue	Setzt den Wert der Eigenschaft zurück.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

GetFieldByIndex-Methode

Sucht das Feld mit dem angegebenen Index.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.GetField (index)
```

Parameter

index
Typ: `Integer`
Nullbasierter Feldindex.

Rückgabetyp

`LSOffice.Field`

Hinweis

Diese Methode gibt eine Referenz auf ein Field-Objekt zurück, oder **null**, falls kein Feld gefunden worden ist.

Siehe auch

- GetFieldByName-Methode
- Label-Klasse
- Objektmodellreferenz

GetFieldByName-Methode

Sucht das Feld mit dem angegebenen Namen.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.GetField (fieldName)
```

Parameter

fieldName
Typ: `String`
Feldname

Rückgabetyp

`LSOffice.Field`

Hinweis

Diese Methode gibt eine Referenz auf ein Field-Objekt zurück, oder **null**, falls kein Feld gefunden worden ist.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Field.GetContent.

Siehe auch

- GetFieldByIndex-Methode
- Label-Klasse
- Objektmodellreferenz

GetPreview-Methode

Erstellt ein Vorschaubild des aktuellen Etiketteninhalts.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.GetPreview ()
```

Rückgabetyp

```
Object
```

Hinweis

Diese Methode gibt eine Referenz auf ein **Bitmap**-Objekt zurück, oder **null**, falls kein Etikett geöffnet ist oder keine Vorschau erstellt werden kann.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

GetPropertyValue-Methode

Gibt den Wert der Eigenschaft zurück.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Verwendung

```
objLabel.GetPropertyValue (propertyName)
```

Parameter

propertyName
Typ: `String`
Eigenschaftenname

Die folgende Tabelle beschreibt einige mögliche Eigenschaftennamen:

Eigenschaftenname	Beschreibung
LabelRotation	Typ: `Integer` 0, 90, 180, 270
LabelType	Typ: `Integer` 0 : Haftetiketten 1 : Endlosetiketten

Rückgabetyp

`Object`

Siehe auch

- Label-Klasse
- Objektmodellreferenz

Print-Methode

Druckt das Etikett.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.Print (copies, [options])
```

Parameter

copies
  Typ: `Integer`
  Druckanzahl

options (optional)
  Typ: `LSOffice.PrintOptions`
  Druckoptionen

Hinweis

Ist *copies* gleich 0, wird das Dialogfeld **Drucken** zur Eingabe der Druckanzahl angezeigt, sonst nicht.

Weitere Informationen und ein ausführliches Beispiel finden Sie unter OLE-Automation -> Erste Schritte.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

PrintToFile-Methode

Druck in Datei.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Verwendung

```
objLabel.PrintToFile (fileName, copies, [options])
```

Parameter

fileName
: Typ: `String`
: Dateiname

copies
: Typ: `Integer`
: Druckanzahl

options (optional)
: Typ: `LSOffice.PrintOptions`
: Druckoptionen

Hinweis

Ist *copies* gleich 0, wird das Dialogfeld **Drucken** zur Eingabe der Druckanzahl angezeigt, sonst nicht.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

Save-Methode

Speichert Änderungen am Etikett.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.Save ()
```

Hinweis

Überprüfen Sie die Eigenschaft Application.LastError um zu sehen ob die Funktion erfolgreich ausgeführt worden ist.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

SaveAs-Methode

Speichert Änderungen am Etikett unter einem anderen Namen.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.SaveAs (path)
```

Parameter

path
  Typ: `String`
  Dateiname

Hinweis

Überprüfen Sie die Eigenschaft Application.LastError um zu sehen ob die Funktion erfolgreich ausgeführt worden ist.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

SavePreview-Methode

Speichert ein Vorschaubild des aktuellen Etiketteninhalts.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Verwendung

```
objLabel.SavePreview (fileName, [format])
```

Parameter

fileName
Typ: `String`
Dateiname

format (optional, Standard = Bmp)
Typ: `LSOffice.ImageFormat`
Dateiformat

Siehe auch

- Label-Klasse
- Objektmodellreferenz

SelectRecord-Methode

Sucht den angegebenen Datensatz.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLabel.SelectRecord (filterExpression)
```

Parameter

filterExpression
Type: `String`
Filterkriterium. Beispiele dazu, wie Datensätze gefiltert werden, finden Sie unter Filtersyntax.

Hinweis

Überprüfen Sie die Eigenschaft Application.LastError um zu sehen ob die Funktion erfolgreich ausgeführt worden ist.

Beispiel (VBScript)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Print Record Sample Code
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Object variables
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim objApp
Dim objLabel

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Constants LSOffice.PrintOptions
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Const PrintOptions_PrintCurrentRecord = 1
Const PrintOptions_PrintAllRecords = 2
Const PrintOptions_Default = 0

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Print record
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If
```

```
' Open label
Set objLabel = objApp.OpenLabel("..\Label3.lbex")

If (objLabel is Nothing) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

' Enter selection string
Dim tmp
tmp = InputBox("Native name starts with:" & vbCrLf & vbCrLf & "de" & vbTab & "Deutschland" &
vbCrLf & "fr" & vbTab & "France" & vbCrLf & "it" & vbTab & "Italia" & vbCrLf & "es" & vbTab &
"Espana" & vbCrLf & "..." & vbCrLf, "Select Record", "de")

If tmp = "" Then
    WScript.Echo "No user input, aborted"
    WScript.Quit
End If

' Select record and print label
objLabel.SelectRecord("NativeName LIKE '" & tmp & "%'")

If (objApp.HasError) Then
     WScript.Echo objApp.LastError.Message
     WScript.Quit
End If

objLabel.Print 1, PrintOptions_PrintCurrentRecord
```

Siehe auch

- Label-Klasse
- Objektmodellreferenz

Filtersyntax

Ein Filter wird mit der Filtersyntaxsprache generiert. Damit können Sie einfache, fortgeschrittene oder erweiterte Filter erzeugen.

Spaltennamen

Enthält ein Spaltennamen eines von diesen Sonderzeichen ~ ( ) # \ / = > < + - * % & | ^ ' " [ ], so muss der Spaltenname in eckige Klammern [ ] eingeschlossen werden. Enthält der Spaltenname ein] oder \, so muss ein Backslash vor die Zeichen gesetzt werden (\] oder \\).

Beispiel	Beschreibung
Filter = "id = 10"	Kein Sonderzeichen im Spaltenname "id".
Filter = "$id = 10"	Kein Sonderzeichen im Spaltenname "$id".
Filter = "[#id] = 10"	Sonderzeichen "#" in Spaltenname "#id".
Filter = "[[id\]] = 10"	Sonderzeichen in Spaltenname "[id]".

Konstanten

Eine Konstante, gelegentlich auch als Literal- oder Skalarwert bezeichnet, ist ein Symbol, das einen bestimmten Datenwert repräsentiert. Das Format einer Konstante ist abhängig vom Datentyp des Werts, den sie repräsentiert.

Zeichenfolgenkonstanten werden in einfache Anführungszeichen ' ' eingeschlossen und enthalten alphanumerische Zeichen (a-z, A-Z und 0-9) sowie Sonderzeichen, wie z. B. Ausrufezeichen (!), @-Zeichen und Nummernzeichen (#). Enthält eine in einfache Anführungszeichen eingeschlossene Zeichenfolge ein eingeschlossenes Anführungszeichen, muss das eingeschlossene Anführungszeichen durch zwei einfache Anführungszeichen ersetzt werden.

Beispiel	Beschreibung
Filter = "Name = 'John'"	Zeichenfolgenkonstante
Filter = "Name = 'John ''A'''"	Zeichenfolgenkonstante mit eingeschlossenem Anführungszeichen "John 'A'".

Zahlenwerte werden durch eine Zeichenfolge von Zahlen dargestellt, die nicht in Anführungszeichen eingeschlossen sind (englische Formatierung verwenden).

Beispiel	Beschreibung
Filter = "Year = 2008"	Ganzzahliger Wert
Filter = "Price = 1199.9"	Float-Wert

Datumskonstanten werden in # # eingeschlossen (englische Formatierung verwenden).

Beispiel	Beschreibung
Filter = "Date = #12/31/2008#"	Datum (Uhrzeit ist 00:00:00).
Filter = "Date = #2008-12-31#"	Auch dieses Format wird unterstützt.
Filter = "Date = #12/31/2008 16:44:58#"	Datum/Uhrzeit

Alternativ können auch alle Werte in einfache Anführungszeichen ' ' eingeschlossen werden. Das bedeutet, dass Textzeichenfolgen für Zahlen und Datum/Uhrzeit verwendet werden können. In diesem Fall wird die aktuelle Ländereinstellung verwendet um die Zeichenfolge in den zugehörigen Wert umzuwandeln.

Beispiel	Beschreibung
Filter = "Date = '12/31/2008 16:44:58'"	Ländereinstellung Englisch

Filter = "Date = '31.12.2008 16:44:58'"	Ländereinstellung Deutsch
Filter = "Price = '1199.90'"	Ländereinstellung Englisch
Filter = "Price = '1199,90'"	Ländereinstellung Deutsch

Vergleichsoperatoren

Vergleichsoperatoren testen, ob zwei Ausdrücke gleichwertig sind. **Vergleichsoperatoren** sind = (gleich), <> (ungleich), < (kleiner), <= (kleiner oder gleich), > (größer) oder >= (größer und gleich).

Hinweis: Beim Vergleich von Zeichenfolgen wird die Groß- und Kleinschreibung beachtet.

Beispiel	Beschreibung
Filter = "Num = 10"	Nummer ist gleich 10.
Filter = "Date < #1/1/2008#"	Datum vor dem 01.01.2008.
Filter = "Name <> 'John'"	Name ungleich John.
Filter = "Name >= 'Jo'"	Zeichenfolgenvergleich

Der **IN-Operator** ermittelt, ob ein angegebener Wert mit einem Wert aus einer Unterabfrage oder Liste übereinstimmt.

Beispiel	Beschreibung
Filter = "Id IN (1, 2, 3)"	Ganzahlige Werte
Filter = "Price IN (1.0, 9.9, 11.5)"	Float-Werte
Filter = "Name IN ('John', 'Jim', 'Tom')"	Textzeichenfolgen
Filter = "Date IN (#12/31/2008#, #1/1/2009#)"	Datum/Uhrzeit
Filter = "Id NOT IN (1, 2, 3)"	Werte nicht in der Liste

Der **LIKE-Operator** bestimmt, ob eine bestimmte Zeichenfolge mit einem angegebenen Muster übereinstimmt. Ein Muster kann normale Zeichen und Platzhalterzeichen enthalten. Bei einem Mustervergleich müssen normale Zeichen exakt mit den angegebenen Zeichen in der Zeichenfolge übereinstimmen. Platzhalterzeichen können jedoch mit beliebigen Teilen der Zeichenfolge übereinstimmen. Das Verwenden der Vergleichsoperatoren für Zeichenfolgen = und <> ist nicht so flexibel wie das Verwenden von Platzhalterzeichen mit dem **LIKE-Operator**.

Platzhalterzeichen sind * oder %. Platzhalterzeichen können am Anfang '*value', am Ende 'value*', oder am Anfang und Ende '*value*' eines Musters eingefügt werden. Platzhalterzeichen in der Mitte eines Musters 'va*lue' sind **nicht erlaubt**.

Beispiel	Beschreibung
Filter = "Name LIKE 'j*'"	Alle Werte fangen mit 'j' an.
Filter = "Name LIKE '%jo%'"	Alle Werte enthalten 'jo'.
Filter = "Name NOT LIKE 'j*'"	Alle Werte fangen nicht mit 'j' an.

Enthält ein Muster eines von diesen Zeichen * % [ ], so müssen die Zeichen in eckige Klammern gesetzt werden, z.B. [*], [%], [[] or []].

Beispiel	Beschreibung
Filter = "Name LIKE '[*]*'"	Alle Werte fangen mit '*' an.
Filter = "Name LIKE '[[]*'"	Alle Werte fangen mit '[' an.

Logische Operatoren

Logische Operatoren sind **AND, OR** und **NOT** und testen den Wahrheitswert einer Bedingung. Der Operator NOT hat Vorrang vor AND und AND hat Vorrang vor OR.

Beispiel	Beschreibung
Filter = "City = 'Tokyo' AND (Age < 20 OR Age > 60)"	Alle Menschen die in Tokyo leben und zwischen 20 und 60 Jahre alt sind.
Filter = "City <> 'Tokyo' AND City <> 'Paris'"; Filter = "NOT City = 'Tokyo' AND NOT City = 'Paris'"; Filter = "NOT (City = 'Tokyo' OR City = 'Paris')"; Filter = "City NOT IN ('Tokyo', 'Paris')";	Alle Menschen die nicht in Tokyo oder Paris leben.

Arithmetische und Zeichenfolgenoperatoren

Arithmetische Operatoren sind + (Addition), - (Subtraktion), * (Multiplikation), / (Division) und % (Modulo).

Beispiel	Beschreibung
Filter = "MotherAge - Age < 20"	Alle Menschen mit einer jungen Mutter.
Filter = "Age % 10 = 0"	Alle Menschen mit zehnjährigem Geburtstag.

Zusätzlich gibt es noch einen **Zeichenfolgenoperator**, + (Verkettung).

Parent-Child-Beziehung referenzieren

In einem Ausdruck kann auf eine **übergeordnete Tabelle (parent table)** referenziert werden, indem man dem Spaltenname das Prefix `Parent.` hinzufügt. Auf eine Spalte in einer **untergeordnete Tabelle (child table)** kann zugegriffen werden, wenn man dem Spaltennamen das Prefix `Child.` hinzufügt.

Der Zugriff auf die untergeordnete Spalte (child column) muss in einer Aggreagtfunktion erfolgen, da untergeordnete Beziehungen (child relationships) mehrere Spalten zurückliefern können. Beispiel: `SUM(Child.Price)` liefert die Summe aller Preise in der untergeordneten Tabelle (child table) zurück, die in Zusammenhang mit der Zeile der übergeordneten Tabelle (parent table) stehen.

Hat eine Tabelle mehr als eine untergeordnete Tabelle (child relation), so muss das Prefix den Tabellennamen enthalten. Beispiel: `Child(OrdersToItemsRelation).Price` refrenziert auf die Spalte `Price` in der Tabelle `OrdersToItemsRelation`.

Aggregatfunktionen

Zu den Aggregatfunktionen gehören folgende Funktionen: **SUM, COUNT, MIN, MAX, AVG** (berechnet den arithmetischen Mittelwert), **STDEV** (ermittelt die Standardabweichung bezogen auf eine Stichprobe) and **VAR** (berechnet die Varianz bezogen auf eine Stichprobe).

Beispiel	Beschreibung
Filter = "Salary > AVG(Salary)"	Alle Menschen mit überdurchschnittlichem Gehalt.
Filter = "COUNT(Child.IdOrder) > 5"	Alle Aufträge mit mehr als fünf Artikeln.
Filter = "SUM(Child.Price) >= 500"	Alle Aufträge mit einem Gesamtpreis (Summe aller Artikelpreise) größer oder gleich 500€.

Funktionen

Folgende Funktionen werden unterstützt. Weitere Informationen finden Sie unter Filterfunktionen.

- `CONVERT` – Wandelt einen Ausdruck in einen anderen Datentyp um.

- `LEN` – Gibt die Anzahl von Zeichen in einer Zeichnenfolge zurück.
- `ISNULL` – Ersetzt **NULL** durch den angegebenen Ersatzwert.
- `IIF` – Gibt einen von zwei Werten zurück, abhängig davon, ob der boolesche Ausdruck **true** oder **false** ergibt.
- `TRIM` – Entfernt alle führende und nachfolgende Leerzeichen aus einer Zeichenfolge.
- `SUBSTRING` – Gibt eine Zeichenfolge aus der Mitte einer Textzeichenfolge zurück, und zwar ausgehend von einer angegebenen Startposition und Länge.

Siehe auch

- Filterfunktionen
- SelectRecord-Methode
- Label-Klasse
- Objektmodellreferenz

Filterfunktionen

Folgende Funktionen werden unterstützt:

CONVERT

Beschreibung	Wandelt einen Ausdruck in einen anderen Datentyp um.
Syntax	CONVERT (*expression*, *type*)
Argumente	*expression* -- Ausdruck, der umgewandelt werden soll. *type* -- Datentyp, in den der Ausdruck umgewandelt werden soll.
Beispiel	Expression = "Convert (total, 'System.Int32')"

Alle Konvertierungen sind möglich, bis auf die folgenden Ausnahmen: **Boolean** kann nur in **Byte**, **SByte**, **Int16**, **Int32**, **Int64**, **UInt16**, **UInt32**, **UInt64**, **String** und sich selbst umgewandelt werden. **Char** kann nur in **Int32**, **UInt32**, **String**, und sich selbst umgewandelt werden. **DateTime** kann nur in **String** und sich selbst umgewandelt werden. **TimeSpan** kann nur in **String** und sich selbst umgewandelt werden.

LEN

Beschreibung	Gibt die Anzahl von Zeichen in einer Zeichnenfolge zurück.
Syntax	LEN (*expression*)
Argumente	*expression* -- Zeichenfolge
Beispiel	Expression = "Len (item)"

ISNULL

Beschreibung	Ersetzt **NULL** durch den angegebenen Ersatzwert.
Syntax	ISNULL (*expression*, *replacementvalue*)
Argumente	*expression* -- Ausdruck, der auf **NULL** überprüft werden soll. *replacementvalue* -- Der Ausdruck, der zurückgegeben werden soll, wenn *expression* **NULL** ist.
Beispiel	Expression = "IsNull (price, -1)"

IIF

Beschreibung	Gibt einen von zwei Werten zurück, abhängig davon, ob der boolesche Ausdruck **true** oder **false** ergibt.
Syntax	IIF (*expression*, *truepart*, *falsepart*)
Argumente	*expression* -- Boolescher Ausdruck, der ausgewertet werden soll. *truepart* -- Dieser Wert wird zurückgegeben, wenn *expression* **true** ergibt. *falsepart* -- Dieser Wert wird zurückgegeben, wenn *expression* **false** ergibt.
Beispiel	Expression = "IIF (total>1000, 'expensive', 'dear')

TRIM

Beschreibung	Entfernt alle führende und nachfolgende Leerzeichen aus einer Zeichenfolge.
Syntax	TRIM (*expression*)
Argumente	*expression* -- Zeichenfolge, aus dem die Leerzeichen entfernt werden sollen.

SUBSTRING

Beschreibung	Gibt eine Zeichenfolge aus der Mitte einer Textzeichenfolge zurück, und zwar ausgehend von einer angegebenen Startposition und Länge.
Syntax	SUBSTRING (*expression*, *start*, *length*)
Argumente	*expression* -- Die Textzeichenfolge, aus der die Zeichen extrahiert werden sollen *start* -- Die Position des ersten Zeichens, das Sie extrahieren möchten. *length* -- Die Anzahl der zurückzugebenden Zeichen.
Beispiel	Expression = "SubString (phone, 7, 8)"

Siehe auch

- Filtersyntax
- SelectRecord-Methode
- Label-Klasse
- Objektmodellreferenz

SetPropertyValue-Methode

Setzt den Wert der Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Verwendung

```
objField.SetPropertyValue (propertyName, value)
```

Parameter

propertyName

Typ: `String`
Eigenschaftenname

Die folgende Tabelle beschreibt einige mögliche Eigenschaftennamen:

Eigenschaftenname	Beschreibung
LabelRotation	Typ: `Integer` 0, 90, 180, 270
LabelType	Typ: `Integer` 0 : Haftetiketten 1 : Endlosetiketten

value

Typ: `Object`
Neuer Eigenschaftenwert.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

LicenseInfo-Klasse

Stellt Eigenschaften zum Abrufen der Lizenzinformationen bereit.

Eigenschaften

	Name	Beschreibung
	IsTrialVersion	Ruft den Wert ab, der angibt, ob die Lizenz eine Testlizenz ist.
	LicenseKey	Gibt den Lizenzschlüssel zurück, mit dem **Labelstar Office** aktiviert worden ist.
	LicenseType	Gibt den Lizenztyp zurück.

Siehe auch

- Objektmodellreferenz

LicenseInfo-Eigenschaften

Dieses Objekt hat die folgenden Eigenschaften.

Eigenschaften

	Name	Beschreibung
	IsTrialVersion	Ruft den Wert ab, der angibt, ob die Lizenz eine Testlizenz ist.
	LicenseKey	Gibt den Lizenzschlüssel zurück, mit dem **Labelstar Office** aktiviert worden ist.
	LicenseType	Gibt den Lizenztyp zurück.

Siehe auch

- LicenseInfo-Klasse
- Objektmodellreferenz

IsTrialVersion-Eigenschaft

Ruft den Wert ab, der angibt, ob die Lizenz eine Testlizenz ist. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLicense.IsTrialVersion
```

Typ

```
Boolean
```

Hinweis

In der Testversion wird zu jeder Grafik ein Wasserzeichen hinzugefügt und es werden alle 'e' durch 'x' und alle '5' durch '0' ersetzt.

Siehe auch

- LicenseInfo-Klasse
- Objektmodellreferenz

LicenseKey-Eigenschaft

Gibt den Lizenzschlüssel zurück, mit dem **Labelstar Office** aktiviert worden ist. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLicense.LicenseKey
```

Typ

```
String
```

Siehe auch

- LicenseInfo-Klasse
- Objektmodellreferenz

LicenseType-Eigenschaft

Gibt den Lizenztyp zurück. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objLicense.LicenseType
```

Typ

```
String
```

Hinweis

Mögliche Lizentypen sind **TRIAL**, **LITE**, **BASIC** oder **PROFESSIONAL**.

Weitere Informationen zu den verschiedenen Lizenztypen finden Sie unter Programmvarianten.

Siehe auch

- LicenseInfo-Klasse
- Objektmodellreferenz

PrintOptions-Enumeration

Die **PrintOptions**-Enumeration legt Optionen für die Print-Methode einer Label-Komponente fest.

Diese Enumeration erlaubt eine bitweise Kombination der Memberwerte.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Member

	Name	Wert	Beschreibung
	Default	0 (0x00)	Standardeinstellungen
	PrintAllRecords	1 (0x01)	Druckt alle Datensätze. Das Dialogfenster **Datensatz auswählen** wird nicht angezeigt.
	PrintCurrentRecord	2 (0x02)	Druckt den aktuell ausgewählten Datensatz. Das Dialogfenster **Datensatz auswählen** wird nicht angezeigt.
	ShowPrintingDialog	4 (0x04)	Zeigt einen **Druckdialog** am, solange Etiketten an den Drucker gesendet werden.
	ShowNotificationMessage	8 (0x08)	Zeigt eine Benachrichtigung im unteren rechten Bildschirmbereich an, wenn das Etikett gedruckt wird.

Hinweis

Weitere Informationen und ein ausführliches Beispiel finden Sie unter Label.SelectRecord.

Siehe auch

- Label-Klasse
- Objektmodellreferenz

VersionInfo-Klasse

Stellt Eigenschaften zum Abrufen von Informationen über die API bereit, z. B. die Versionsnummer, Copyright, usw.

Eigenschaften

	Name	Beschreibung
	CompanyName	Ruft den der API zugeordneten Firmennamen ab.
	CompiledVersion	Interne Versionsnummer der API (dieses Feld ist nur für den internen Gebrauch).
	Copyright	Ruft den der API zugeordneten Copyrightvermerk ab.
	DisplayVersion	Ruft die Versionsnummer der API ab.
	ProductName	Name des Produkts.

Siehe auch

- Objektmodellreferenz

VersionInfo-Eigenschaften

Dieses Objekt hat die folgenden Eigenschaften.

Eigenschaften

	Name	Beschreibung
	CompanyName	Ruft den der API zugeordneten Firmennamen ab.
	CompiledVersion	Interne Versionsnummer der API (dieses Feld ist nur für den internen Gebrauch).
	Copyright	Ruft den der API zugeordneten Copyrightvermerk ab.
	DisplayVersion	Ruft die Versionsnummer der API ab.
	ProductName	Name des Produkts.

Siehe auch

- VersionInfo-Klasse
- Objektmodellreferenz

CompanyName-Eigenschaft

Ruft den der API zugeordneten Firmennamen ab. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objVersion.VendorName
```

Type

```
String
```

Beispiel

CompanyName: "Carl Valentin GmbH"

Siehe auch

- VersionInfo-Klasse
- Objektmodellreferenz

CompiledVersion-Eigenschaft

Interne Versionsnummer der API (dieses Feld ist nur für den internen Gebrauch). Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objVersion.CompiledVersion
```

Typ

```
String
```

Beispiel

CompiledVersion: "4.10.1010"

Siehe auch

- VersionInfo-Klasse
- Objektmodellreferenz

Copyright-Eigenschaft

Ruft den der API zugeordneten Copyrightvermerk ab. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objVersion.Copyright
```

Typ

```
String
```

Beispiel

Copyright: "Copyright © Carl Valentin GmbH, 2012-2014"

Siehe auch

- VersionInfo-Klasse
- Objektmodellreferenz

DisplayVersion-Eigenschaft

Ruft die Versionsnummer der API ab. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objVersion.DisplayVersion
```

Typ

```
String
```

Beispiel

DisplayVersion: "Version 4.10 Build 1010"

Siehe auch

- VersionInfo-Klasse
- Objektmodellreferenz

ProductName-Eigenschaft

Name des Produkts. Nur-Lese-Eigenschaft.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Verwendung

```
objVersion.ProductName
```

Typ

```
String
```

Beispiel

ProductName: "Labelstar Office"

Siehe auch

- VersionInfo-Klasse
- Objektmodellreferenz

Fehlercodes

Fehlerbehandlung

Bei der Fehlerbehandlung ist es wichtig zu wissen was den Fehler ausgelöst hat. Wurde der Fehler von einer OLE-Automation Methode oder Eigenschaft ausgelöst können Sie detaillierte Fehlerinformationen erhalten indem Sie die Eigenschaft Application.LastError abfragen. Jeder Aufruf einer OLE-Automation Methode oder Eigenschaft (ausser LastError) setzt die Fehlerinformationen zurück, so dass Application.LastError immer die Fehlerinformationen für die zuletzt aufgerufene Methode oder Eigenschaft enthält.

Fehlercodes

Error Code	Beschreibung	Type
1000	Generischer Fehler.	Fehler
1001	Die Anwendung ist bereits initialisiert.	Warnung
1002	Ungültiger Lizenzschlüssel.	Warnung
1003	Ungültiger Lizenztyp; Professional Lizenz erwartet.	Warnung
1004	Die Anwendung ist nicht initialisiert; rufen Sie zuerst Initialize() auf.	Fehler
1005	Es ist kein Etikett geöffnet.	Fehler
1006	Ungültiger Feldname.	Fehler
1007	Kein Feld definiert.	Fehler
1008	Der Feldindex ist außerhalb des gültigen Bereichs.	Fehler
1009	Ungültiger Eigenschaftenname.	Fehler
1010	Keine Datenbankfelder auf dem Etikett definiert.	Fehler
1011	Kein Datensatz gefunden.	Fehler
1012	Mehrere Datenbankverbindungen auf dem Etikett.	Fehler

Beispiel (VBScript)

```

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Open and print label
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Set objApp = CreateObject("LSOffice.Application")

' Application must be initialized before OpenLabel is called
objApp.Initialize()

If (objApp.HasError) Then
    WScript.Echo objApp.LastError.Message
    WScript.Quit
End If

' Browse file name
Dim fileName
fileName = objApp.GetOpenFilename("Labels|*.lbex|All Files|*.*")

If (Len(fileName) = 0) Then
    WScript.Quit
End If

' Open label
Set objLabel = objApp.OpenLabel(fileName)

If (objLabel is Nothing) Then
    WScript.Echo objApp.LastError.Message
    WScript.Quit
End If

' Print label
objLabel.Print(1)

```

Programmvarianten

Labelstar Office ist in drei Varianten verfügbar. In der LITE-Version eignet sich die Software vor allem für die Gestaltung einfacher Etiketten. Für professionelle Anforderungen gibt es die BASIC- oder PROFESSIONAL-Version. Hier steht eine breite Auswahl an Formaten und Variablen zur Verfügung, so können Etikettieranforderungen aus nahezu allen industriellen Branchen bedient werden.

	LITE	BASIC	PROFESSIONAL
Texte			
TrueType-Schriften	●	●	●
Druckerschriften		●	●
Textformatierung (Markup-Tags)		●	●
Gebogener Text			●
Barcodes			
1D Barcodes	●	●	●
2D Barcodes		●	●
GS1 Barcodes		●	●
Grafiken	Eingeschränkt (nur BMP)	Über 90 Grafik- und Vektorformate (z. B. TIFF, GIF, JPEG, PNG, WMF, BMP, ICO ...)	●
Variablen			
Systemvariablen	Eingeschränkt (nur Datum, Uhrzeit, Numerator und Benutzereingabe)	Über 30 Variablen (z. B. Datum, Uhrzeit, Numerator, Benutzereingabe, Kettenfeld, Prüfziffer, If..Then..Else-Anweisung)	Komplexe Variablendefinitionen (z.B. benutzerdefinierte Prüfziffernberechnung)
Druckervariablen		●	●
Datenbanken		●	●
Protokollierung			●
Memory Card-Unterstützung		●	●
Symbole		●	●
Drucken			
Internes Druckerprotokoll (CVPL) (Carl Valentin Druckertreiber Version 2.3.1 oder höher)		●	●
Druckvorschau		●	●
Druckeinstellungen		●	●
Zweifarben-Druck		●	●
Print-Only			●
OLE-Automation			●
Import Labelstar PLUS Etiketten		●	●

Lizenzierung

Die nachstehenden Informationen sollen Ihnen dabei helfen, ihr Programm zu aktivieren. Falls hierbei Probleme auftreten sollten, kontaktieren Sie bitte den **Labelstar Office** Support.

Wie aktiviere ich mein Programm?

Mit Hilfe des **Lizenzierungs-Assistenten** können Sie ihr Programm aktivieren. Dazu benötigen Sie eine Lizenznummer die Sie auf einem Lizenzetikett in ihrer Programm CD finden.



Wie kann ich feststellen, ob meine Software bereits aktiviert wurde?

1. Öffnen Sie **Labelstar Office**.
2. Klicken Sie auf **Datei** > **Hilfe**.
3. Unter **Info zu Labelstar Office** werden die Produktinformationen angezeigt.

Hinweis: Um weitere Informationen zur Lizenzierung zu erhalten, klicken Sie auf **Weitere Versions- und Copyrightinformationen**. Der **Info über-Dialog** wird geöffnet. Klicken Sie auf die Taste **Programminformationen** und wählen Sie dann **Lizenzierung** aus.

Was ist eine Testversion?

Eine Testversion ermöglicht es Ihnen das Programm zu testen. In der Testversion sind einige Funktionen eingeschränkt, alle e werden durch x und alle 0 durch 5 ersetzt und alle Bilder werden mit einem Wasserzeichen versehen. In der Testversion sind möglicherweise bestimmte Funktionen oder Programme aktiviert, die nicht im Lieferumfang des von ihnen gekauften Produktes enthalten sind. Nachdem Sie eine gültige Lizenznummer eingegeben haben, werden nur die von ihnen gekauften Programme und Features angezeigt.

Was ist unter "Umwandeln" zu verstehen?

Sie können eine Lizenznummer löschen um sie z.B. auf einem anderen Rechner verwenden zu können. Nach dem Löschen der Lizenznummer wird das Programm als Testversion ausgeführt. Sie können auch eine andere Lizenznummer eingegeben um zusätzliche Features und Programme freizuschalten.

Software Update

Um ein Labelstar Office Update durchzuführen, gehen Sie bitte folgendermaßen vor:

1. Öffnen Sie **Labelstar Office**.
2. Klicken Sie auf **Hilfe** in der Registerkarte **Datei** und anschließend auf **Nach Updates suchen.**
   Der **Update-Assistent** wird geöffnet.
3. Folgen Sie den Anweisungen im Assistenten.

oder besuchen Sie unsere Updates-Webseite um die letzte Programmversion herunterzuladen.

Kontakte

Produkt-Webseite

Zusätzliche Informationen zu **Labelstar Office** und die aktuellste Programmversion finden Sie auf unserer Webseite:
www.carl-valentin.de

E-Mail

Technischer Support: support@carl-valentin.de
Bestell- und Lizenzierungsanfragen: order@carl-valentin.de
Allgemeine Anfragen: info@carl-valentin.de

Systemanforderungen

Minimale Systemanforderungen

- Microsoft Windows 7/8/8.1 x86/x64
- .Net Framework 4.0 oder höher (Download unter http://www.microsoft.com/net/)
- Microsoft Visual C++ 2010 Redistributable (x86)
- Microsoft Access Database Engine 2010 (x86)
- Empfohlene Druckertreiber: Carl Valentin Druckertreiber Version 2.3.1 oder höher

Impressum

Carl Valentin GmbH
Neckarstrasse 78-86 u. 94
78056 Villingen-Schwenningen

Telefon: +49 (0) 7720 9712 - 0
E-Mail: info@carl-valentin.de

Copyright © 2015 Carl Valentin GmbH

Haftungsausschluss

Die Beschreibungen in diesem Handbuch stellen keine zugesicherten Eigenschaften im rechtlichen Sinne oder im Sinne der Produkthaftung dar. Die Autoren behalten sich das Recht vor, an der Software Änderungen vorzunehmen, ohne Verpflichtung diese Änderungen irgendeiner Person bekanntzugeben. Es wird keine Garantie für die Richtigkeit es Inhalts dieses Handbuchs übernommen. Da sich Fehler trotz aller Bemühungen nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.

Warenzeichenhinweise

Alle in diesem Handbuch erwähnten Produktbezeichnungen können Warenzeichen oder eingetragene Warenzeichen der entsprechenden Inhaber sein.